NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

THEATER AIR APPORTIONMENT AND ALLOCATION: APPLICATION OF DYNAMIC ALGORITHMS FOR COMBAT MODELS

by

Jeffrey P. Hamman

December, 1995

Thesis Advisor: Mark A. Youngren

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	December 1995	Master's Thesis

4. TITLE AND SUBTITLE
THEATER AIR APPORTIONMENT AND ALLOCATION: APPLICATION OF DYNAMIC ALGORITHMS FOR COMBAT MODELS

5. FUNDING NUMBERS

6. AUTHOR(S)
Hamman, Jeffrey P.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release, distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The purpose of this thesis is to design, develop, and demonstrate a decision model to dynamically apportion and allocate air assets in a theater level combat model subject to specified air campaign plan. The methods described are combined into a stand-alone air apportionment / air allocation $(AA)^2$ model.

The model translates the objectives of an air campaign phases into sets of strength categories that are used to value enemy unit capabilities. These strength categories are assigned desired levels by a user that define when a strength has been reduced to an acceptable level. These desired levels apportion friendly air assets between different strength categories based on the objectives the campaign phases. The strength categories are also used to value each potential target. These values are used to allocate sorties to reduce the strength of the potential target to the desired level. The model uses the strength values of potential targets to determine if the objectives of a campaign phase have been satisfied and whether to activate any follow-on phases. A demonstration of the $(AA)^2$ model is included using a two phase air campaign.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Air apportionment, Air allocation, JFACC, JWAEP, Combat models	121
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18
298-102

Approved for public release; distribution is unlimited.

THEATER AIR APPORTIONMENT AND ALLOCATION: APPLICATION OF DYNAMIC ALGORITHMS FOR COMBAT MODELS

Jeffrey P. Hamman
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL

December 1995

Author: _____
Jeffrey P. Hamman

Approved by: _____
Mark A. Youngren, Thesis Advisor

_____
Samuel H. Parry, Second Reader

_____
Frank C. Petho, Chairman
Department of Operations Research

ABSTRACT

The purpose of this thesis is to design, develop, and demonstrate a decision model to dynamically apportion and allocate air assets in a theater level combat model subject to specified air campaign plan. The methods described are combined into a stand-alone air apportionment / air allocation $(AA)^2$ model.

The model translates the objectives of an air campaign phases into sets of strength categories that are used to value enemy unit capabilities. These strength categories are assigned desired levels by a user that define when a strength has been reduced to an acceptable level. These desired levels apportion friendly air assets between different strength categories based on the objectives the campaign phases. The strength categories are also used to value each potential target. These values are used to allocate sorties to reduce the strength of the potential target to the desired level. The model uses the strength values of potential targets to determine if the objectives of a campaign phase have been satisfied and whether to activate any follow-on phases. A demonstration of the $(AA)^2$ model is included using a two phase air campaign.

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The purpose of this thesis is to design, develop, and test in principle a decision process to apportion and allocate tactical air assets in support of theater campaign objectives. This decision process is demonstrated in the thesis as a stand-alone decision model called the Air Apportionment / Air Allocation $(AA)^2$ model and is intended for use by campaign planners in deciding how changes in desired objectives effect the overall apportioning and allocating of air assets. The $(AA)^2$ model's data structure is based on the structure of the Joint Warfare Experimental Analysis Prototype (JWAEP) being developed at the Naval Postgraduate School. The $(AA)^2$ model's methods are intended for future inclusion into JWAEP.

The decision process used to apportion and allocate air assets based on user defined campaign objectives is divided into two main levels. The first level (apportionment) determines the apportionment of air assets between different enemy strength categories based on the objectives of each phase of an air campaign. The second level (allocation) determines the optimal allocation of air assets to targets based on the apportionment values determined in the first level. The methods for each level are repeated for each day of the campaign with the results showing how the apportionment decisions effect air asset allocation.

In level one, the objectives of a campaign are translated into units usable by the allocation level of the $(AA)^2$ model. This translation is made by first grouping the objectives of on air campaign into one or more sequential phases. Each objective is associated with one or more enemy strength categories that are to be reduced by air attacks. Each strength category selected is assigned a value (desired level) by the user defining what reduction of the enemy's strength in the category is desired. These in turn are used to determine what reduction in strength is required for each strength category to satisfy the objectives of a campaign phase. The resultant values are used by level two to

optimally allocate air assets to achieve the desired results. Completion criteria are defined in the model which determine if a phase's strength category desired levels have been reached to a satisfactory level to begin the next phase of the campaign.

Level two uses a mixed integer linear program to allocate available air assets based on the desired levels determined by the apportionment decision made in level one. The output of this level is a listing of missions to be flown by tactical air assets and the desired amount of fighter and electronic warfare aircraft for protection from enemy air defenses.

This model gives campaign planners a decision tool to analyze how changes in phasing of air operations and different reduction in enemy strengths effect the outcome of an air campaign.

I. INTRODUCTION

United States joint military doctrine designates a Joint Force Commander (JFC), who is responsible for synchronizing the actions of forces from different services assigned to a theater of operations. National or regional strategic objectives guide the JFC in developing a campaign plan that details the major operations to be conducted and their expected time of completion. The campaign plan is made up of operational level objectives which, when satisfied, will result in the accomplishment of the strategic objectives. The JFC uses the campaign plan to guide his subordinate commanders in employment of their forces to maximize the campaign's likelihood of success. In many situations the JFC will partition the overall theater campaign plan into mutually supporting plans for air, land, and naval forces. Each of these plans will contain objectives that are to be achieved by the respective force types. A campaign plan is often partitioned into phases that contain several objectives achieved through the use of mutually supporting operations.

The JFC and his subordinate Joint Force Air Component Commander (JFACC) develop an air campaign plan to apportion and allocate air assets in support of the theater campaign plan objectives. The objectives of a phase of the air campaign plan determine the priority of air missions to fly. The extent to which a phase's objectives have been satisfied determines the proportion of assets to assign in support of the current phase and to any subsequent phase. As the theater campaign transitions from one phase to the next, mission priorities of the air campaign plan change to satisfy the next phase's objectives. Since air assets can carry out numerous types of missions important in accomplishing a campaign's objectives, the JFC and subordinate commanders pay close attention to their apportionment and allocation. As a result, the development of appropriate decision logic to accurately represent this process in combat models is needed.

The objective of this thesis is to build a decision model to dynamically apportion and allocate air assets in a theater level combat model, subject to available assets and a

specified air campaign plan. The model consists of algorithms to determine the optimal apportionment and allocation of available air assets to different types of enemy targets. It deals specifically with air-to-surface attack missions, but also provides a framework for other types of missions. The methods are appropriate for both U.S. and non-U.S. air doctrine and may be used for opposing sides in combat simulations. The methods are combined into a stand-alone air apportionment/air allocation $(AA)^2$ model, which is slated for future inclusion in the Joint Warfare Analysis Experimental Prototype (JWAEP) model currently under development at the Naval Postgraduate School.

Chapter II contains the background information necessary to describe how air assets are apportioned and allocated in a theater level conflict. Current joint doctrine is used to describe the command structure used by the JFC and the responsibilities of the JFACC and other component commanders. A brief explanation of how a JFC develops a theater campaign plan, partitions it into phases and uses it to develop an air campaign plan for apportioning and allocating air assets is provided.

Chapter III gives a brief description of how the air asset apportionment and allocation processes are utilized in current theater level models. The methods used by the TACWAR and THUNDER theater models are discussed. The chapter ends with a brief description of the JWAEP architecture and values generated by it that are used in the $(AA)^2$ model.

Chapter IV describes the methodology of the $(AA)^2$ model. The chapter begins with a discussion of how potential targets are valued based on different strengths that they possess. This is followed by a description of how the objectives of each phase of the air campaign are translated into values used to apportion air assets. These values are used by the allocation phase of the model to allocate air sorties to missions and targets. The chapter ends with a discussion of how the $(AA)^2$ model transitions from one air campaign phase to the next.

Chapter V describes how the $(AA)^2$ model is implemented and gives a practical example using data similar to what would be obtained from the JWAEP model. This

example considers a two phase air campaign using tactical air assets against enemy ground targets.

Chapter VI discusses the assumptions made during the formulation of the $(AA)^2$ methods. A brief discussion of future areas of research concerning apportionment and allocation of air assets concludes the chapter.

II. JOINT OPERATIONS

The United States military doctrine has undergone considerable restructuring to emphasize the combining of components of different services into a joint force under one commander. Joint doctrine designates a Joint Force Commander (JFC) responsible for organizing and synchronizing the operations of the various service components within a theater of operations. The lead document for U.S. joint operations gives a description of the advantages of joint warfare.

> Joint operations doctrine reflects the nature of modern warfare and the strategic requirements of our nation. It is built on a sound base of warfighting theory and practical experience. It seeks to provide JFCs with a broad range of options to defeat an enemy in war or to conduct operations other than war. It is a doctrine that recognizes the fundamental and beneficial effects of teamwork and unity of effort, and the synchronization of military operations in time, space, and purpose. The first fundamental for employment of U.S. joint forces is to achieve strategic aims as rapidly as possible, with the least possible loss of American lives. [Ref. 1]

A. ORGANIZATION OF FORCES

The JFC is responsible for organizing and guiding the forces of different services assigned to him within a theater of operations. Joint doctrine states that "JFCs have full authority to assign missions, redirect efforts, and direct coordination among subordinate commanders." [Ref. 1] The first principle of joint force organization is that the JFC organizes his forces to accomplish the assigned mission based on his vision and concept of operations. This organization takes into account unity of effort, centralized planning, and decentralized execution of joint forces to take advantage of their versatility, responsiveness and initiative. The JFC determines how a joint force is organized by determining the command relationships between service component commanders and functional component commanders. The JFC may have several service component commanders with their respective forces assigned to him. Functional component

commanders are designated by the JFC to combine the capabilities of assets from different services under one commander. Examples of functional component commanders include the Joint Force Land Component Commander (JFLCC), Joint Force Air Component Commander (JFACC), and Joint Force Maritime Component Commander (JFMC). These components are display in Figure 1.

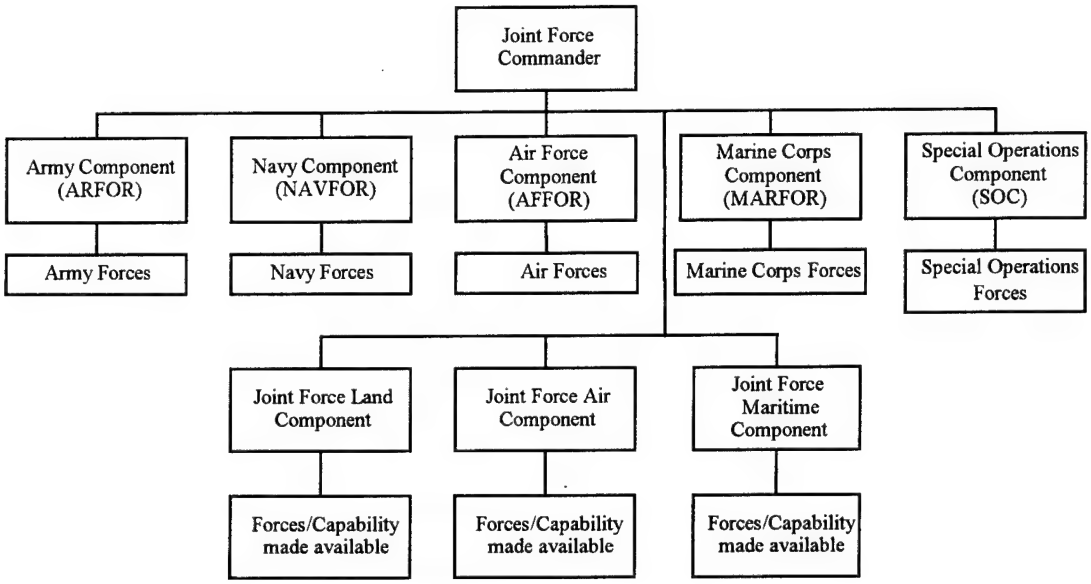

Figure 1. Possible components in a Joint Force.

The JFC will normally designate functional component commanders when there are large numbers of assets from several services within the theater. For example, if Air Force, Navy, and Marine Corps air units are assigned to a theater, the JFC may give control of the assets to a JFACC. This gives the JFC a single component commander to coordinate the employment of air assets throughout the theater. The versatility and flexibility of air assets in support of various operations normally give reason for the JFC to designate control of air assets within the theater to a JFACC. During Operation Desert Shield and Storm all U.S. and coalition air forces were put under the command of a JFACC. The JFACC, LTG Charles Horner, USAF, later wrote, "Jointness afforded us the

opportunity to capitalize on our capabilities without losing service identities. Placing all air forces under the command of the JFACC was a successful application of our military doctrine." [Ref. 2] The actual command relationships depend on the different services assigned to the JFC, and which service has the majority of forces within each functional group. This thesis assumes that the JFC has designated a JFACC and all air assets are available for tasking by him.

B. CAMPAIGN PLANS

A successful joint operation is based on the transformation of national or regional strategic objectives into theater operational objectives. A JFC uses what is referred to as "operational art" to link the strategic objectives to the operational objectives. "Operational art determines when, where, and for what purpose major forces will be employed....It governs the deployment of those forces, their commitment to or withdrawal from battle, and the arrangement of battles and major operations to achieve operational and strategic objectives." [Ref. 1]

One method used by a JFC to develop theater objectives from strategic objectives is to answer the following questions:

- What military conditions are needed to achieve the strategic goal? (Ends)
- What sequence of actions is most likely to produce the conditions? (Ways)
- How should the resources of the joint force be applied to accomplish that sequence of actions? (Means)
- What are the likely costs or risks to the joint force in performing that sequence of actions? (Risks)

The JFC answers these questions to help formulate a campaign plan that defines the sequence of actions required (Ways) to use the designated joint force (Means) in producing the desired strategic goal (Ends) while taking into account the possible costs to

the force (Risks). A JFC uses campaign plans to synchronize the operations of air, land, sea forces in carrying out his concept of operations. The campaign plan describes what, how and when forces combine to carry out operations in achieving the desired objectives within the specified time. "A campaign plan is a plan for a series of related military operations aimed at accomplishing a common objective, normally within a given time and space. It expresses a commander's vision and intent for a developing situation. The purpose of the plan is to convey the strategic decisions made by the commander so that detailed execution planning can proceed on an orderly basis and in sufficient time to assemble the means to achieve the assigned objective." [Ref. 1] The campaign plan is a primary means by which a JFC provides for strategic unity of effort for forces assigned and guides the planning of joint operations within his theater of operations. Campaign plans are used to ensure that subordinate commanders' plans are consistent with the strategy, guidance, and direction developed by the JFC and contribute to achieving theater objectives. The JFC's campaign plan is the guiding concept by which a theater's ground, naval and air campaign plans are formulated. These plans are constructed to achieve the objectives of the overall campaign plan using the respective forces assigned to the subordinate commander. Each of these subordinate campaign plans is both dependent and supportive of the other plans. For example, the ground campaign plan may be dependent on the air campaign plan obtaining air superiority over the enemy before a ground offensive can take place.

The arrangement of major operations within a campaign plan relates directly to the JFC's decision on phasing. A phase represents a period during which a large portion of the forces are involved in similar or mutually supporting activities. Phases group several objectives of the campaign together so that the joint forces can more effectively apply their abilities to achieve objectives. Transitions from one phase to the next represent a shift in emphasis from one set of objectives to the next. The primary benefit of phasing is that it assists the JFC in achieving major objectives which cannot be attained all at once. This is done by planning manageable subordinate operations to gain progressive advantages that

together achieve the major objectives as quickly and affordably as possible. The phases of a campaign can be carried out sequentially or concurrently depending on how many resources are available and the success of previous operations. Phases may overlap and the point where one phase stops and another begins is often difficult to define in absolute terms. Examples of how different phases may be arranged are shown in Figures 2, 3, and 4. During planning, commanders establish conditions for transition from one phase to another. A phase is considered completed when all the conditions have been satisfied to a specified degree. No campaign plan can project with confidence much beyond the initial stages of the operation. Branches and sequels allow the JFC to build flexibility into the campaign plan to preserve freedom of action in rapidly changing conditions. A branch is a campaign phase not in the original campaign plan that is executed in place of another phase if certain campaign conditions are met. Branches may include a shift in target priorities, movement of forces, or a change in campaign objectives as a result of enemy action, availability of friendly assets, or even a change in the weather within the operational area. Sequels are subsequent phases that are executed based on the possible outcomes of the current phase of operations. A campaign plan that is executed sequentially is in principle a series of sequels.

C. APPORTIONMENT AND ALLOCATION

Apportionment is defined by joint doctrine as the assignment of effort by percentage or priority to devote to the various air operations for a given period. The JFC makes this decision to ensure that the focus of the air effort is consistent with campaign phases and objectives. The dilemma for the JFC is to determine the proper balance between different types of air missions at any given point during a campaign. Given the many functions that air power can perform, its theater-wide application, and its ability to rapidly shift from one function to another, the JFC must pay particular attention to its apportionment. The JFC can also use the apportionment decision to synchronize air, land and naval assets in support of the campaign plan's timing. The objectives of the current

phase strongly influence any apportionment decision. A transition from one phase to the next results in a shift of the air apportionment values.

Once the JFC has made an apportionment decision, the JFACC allocates available air assets to missions in support of the decision. The JFACC's challenge is to optimally allocate air assets to missions that best support the JFC's apportionment values and campaign objectives. To do this the JFACC prioritizes all available targets and missions based on their importance to the current phase of the campaign. The JFC's apportionment decision gives direction to the types of missions and targets to assign air assets. The JFACC then prioritizes the missions and targets and allocates air assets to them. Depending on the way that a campaign plan is executed, attacks of different target sets may take place in series or in parallel. A campaign plan executed sequentially results in all the components of the highest priority target set being attacked first before attacks initiated on the next target set. Attack in parallel refers to attacking targets across several different target sets at the same time.

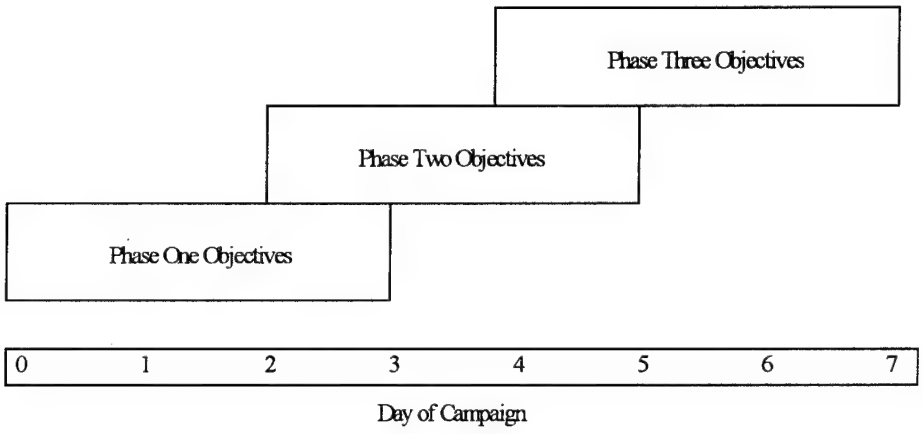

Figure 2. Mixed phasing.

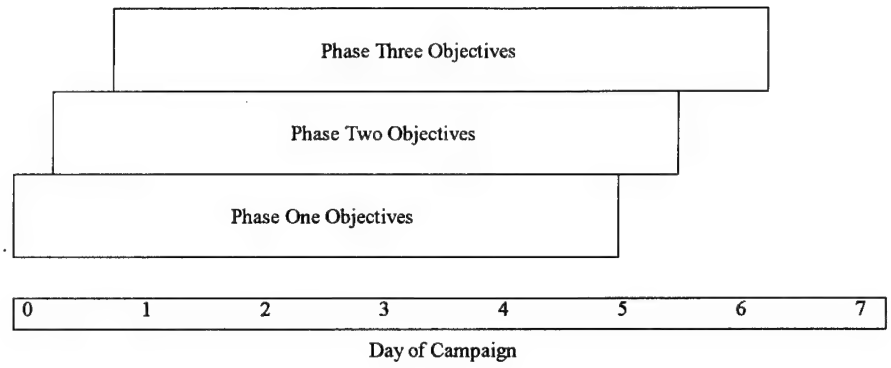

Figure 3. Concurrent phasing.

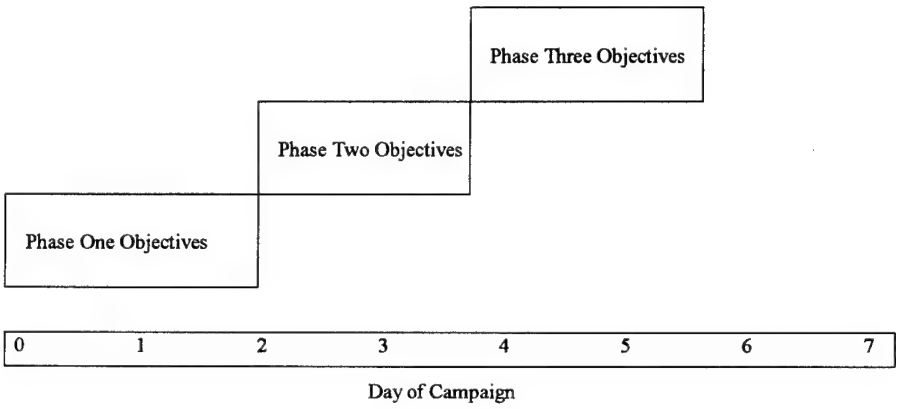

Figure 4. Sequential phasing.

D. TARGET VALUE

The value of possible targets for air attack is dependent on the apportionment decision made by a JFC and the current situation within the theater. Apportionment determines what types of missions and targets are to be attacked according to the phase of the campaign plan. The priority among missions and targets is determined by the value each possesses in accomplishing the stated objectives of the phase. Two like targets may be valued differently depending on their locations, current strength and value to the opposite side. The highest priority targets are ones whose destruction or neutralization

results in the greatest benefit in accomplishing a stated objective. The JFACC must determine the value of each mission and target to optimally allocate available resources. "The targeting process is cyclic. It begins with guidance and priorities issued by the JFC and continues with identification of requirements by components, the prioritization of these requirements, the acquisition of targets or target sets, the attack of targets by components, the assessment of the effects of those missions by both the components and JFC, and continuing guidance from the JFC on future fires or attacks of targets." [Ref. 1] An important part of the targeting process is the availability and accuracy of information of opposite side assets, movements and intentions. Knowing an enemy's intention simplifies the targeting process by identifying those enemy elements whose targeting will result in the greatest reduction in enemy capability. Because of this, a great deal of effort is expended by theater assets in gathering information for the targeting process. Typically, JFCs organize Joint Targeting Coordination Boards (JTCB) to review target information, develop targeting guidance and priorities, and prepare and refine joint target lists. The JTCB consists of members from each of the services that recommend targets and missions that support his respective service or component. If established, the JTCB gives each component within the theater a forum for nominating targets or missions they believe important to accomplishing their stated objective. The product of the apportionment, allocation and target prioritization process is an Air Tasking Order (ATO). The ATO specifies for a given period the missions to be flown by air assets. The ATO is the product of the JFC's campaign plan, apportionment decision and the JFACC's allocation of air assets. It represents the tactical operations required to accomplish the theater strategic objectives.

III. MODEL COMPARISONS

This chapter compares the methods used by the TACWAR and THUNDER theater models in apportioning air assets and generating a prioritized target list. Also included is a brief discussion of the data architecture and methods used by the Joint Warfare Analysis Experimental Prototype (JWAEP) model.

A. TACWAR MODEL

The TACWAR model is a deterministic two-sided theater-level model that simulates both ground and air combat. [Ref. 3] The model's battle area is divided into sectors, regions, and partitions. Sectors are corridors that ground forces maneuver within. Regions are adjacent sectors that are grouped together: Aircraft are restricted from flying between regions. These regions are further divided into region partitions. The partition of regions is based on distance from the Forward Edge of the Battle Area (FEBA). An analyst inputs what aircraft are located in each region, to which region partition they are capable of flying, and what missions they can perform. Six air missions are defined including air base attack, interdiction, close air support, surface-to-air missile (SAM) suppression, battlefield defense, and area defense. The analyst is required to assign the fraction of each type of aircraft assigned to each mission type for each region partition. The fractions are used to determine the number of available sorties from each air base for each region partition and mission type. These assignments remain constant throughout the run and are unaffected by the outcome of either ground or air combat. For each mission type, the analyst also assigns the enemy units and components to be targeted by the attacking aircraft. No prioritization of targets exists other than what is entered by the analyst. Sorties using the user-input mission assignments continue to be allocated to a unit as long as the unit is considered active. TACWAR contains no dynamic structure to alter these values based on the model outcomes.

B. THUNDER MODEL

The Air Force Studies and Analyses Agency (AFSAA) THUNDER model is a two-sided theater-level model that represents both air and ground combat. The ground model has most of the features described for TACWAR. For the air war, sectors are broken into zones, similar to TACWAR's region partitions. THUNDER does not restrict aircraft from flying across sector boundaries, giving them freedom of movement to attack any targets within their range. The apportionment of air assets within the THUNDER model is set by the analyst by assigning values to 21 different air mission types. [Ref. 4] These values are used to generate the number of sorties available to each mission type. The analyst also defines the effectiveness of each aircraft type for each mission type.

All units modeled in THUNDER are considered as potential targets for air sorties. The generation of a prioritized list of targets for air attacks can be computed using one of three methods. The first method (manual mode) allows the analyst to set the priority of each target. In the second method (automatic mode), the model calculates the priority of each potential target. The third method uses the target priorities calculated by the second method, then multiplies them by a user defined zone sector value. This allows targets in specified sectors to be given a higher priority than others. In the last two methods, the priority of each target is determined by prioritization functions. There are functions for each mission type; potential targets are assigned to one or more functions. The functions take into account several factors, including the relative strength, location, and type of each potential target, to calculate a target value. The resultant values are used to generate a prioritized target list for use in allocating air assets to targets. Like TACWAR, THUNDER does not dynamically apportion air assets, but rather relies on analyst input.

C. JWAEP

As was mentioned earlier, the decision model developed in this thesis is for future inclusion in the JWAEP model. JWAEP extends earlier work done with the Future

Theater Level Model (FTLM) developed by NPS and the Arc-Node model developed by George Mason University for Argonne Laboratories. [Ref. 5] JWAEP differs from TACWAR and THUNDER in that it is stochastically based rather than deterministic.[1] Since the model outcomes are derived from stochastic processes, the model can produce different outcomes on subsequent model runs using the same input. This produces a range of outcomes that allows the user to weigh the consequences of various events, which leads to a more realistic modeling of actual combat and its uncertainties.

1. Model Architecture

The model's architecture consists of two arc-node networks; one for surface movement and one for air. The surface units represented in the model exist and maneuver within a user defined network representative of the theater of operations. This network has physical nodes and arcs that connect them. Physical nodes represent locations of geographical points of interest, air bases, logistic facilities, critical intersections, operating areas, and assembly areas and arcs exist between two physical nodes to represent the terrain characteristics, size, or mobility characteristics of paths between two nodes. The units represented within JWAEP maneuver by moving from physical node to physical node over arcs. Aircraft operate within a second network that is overlaid above the surface network. This air network is a square grid system whose size and resolution is defined by the user for the area of operations. This structure allows aircraft freedom to maneuver beyond the restrictions of the arc-node system of the surface network. An aircraft flight consists of an aircraft first taking off from an air base located in the surface network, transiting through the air network to a designated target grid, and performing its mission within the bounds of that grid. The assigned mission may include attacking a ground, naval, or air unit, performing reconnaissance on enemy units, or protecting other airborne aircraft from enemy aircraft or air defense systems.

---

[1] The air war in THUNDER is stochastic, but the ground war is deterministic.

2. Unit Types

In JWAEP, each instance of a ground unit, air base, C^3I unit, logistic facility, or naval unit is referred to as a unit. Units are characterized not only by their type but also by what side they are on, their size, and the equipment that is attached to them. Examples include brigades for ground units, flight groups for air units, and ships for naval units. The equipment attached to a unit may include weapon, C^3I, mobility, and logistic systems. The amounts of each are defined in the unit definition using a structure called the Table of Organization and Equipment (TO&E). The TO&E data give the amounts of each type of equipment attached to a standard unit type of a given size.

3. Perceptions

The stochastic architecture of JWAEP allows for the generation and use of perceptions. In the model, perceptions represent friendly and enemy perceived values for the locations, dispositions, and courses of action for the enemy forces. In JWAEP, two definitions are used to identify different perception values related to the current campaign situation. "Ground truth" is the actual location, composition, and movements of both sides' forces currently being used by the model. From these values and the intelligence assets represented, the model calculates "perceived" values that represent, for each side, the perceived locations, compositions and movements of the other side's forces. Each unit instance within JWAEP is subject to detection by the opposite side's sensors. The detections by the sensors are used to build a perception of the number and type of units located at each surface node or arc. Each sensor detection gives the number and type of equipment detected; these data are compared to the TO&E data of the defined unit types to calculate the probability that the sensor detected a specific unit type. These probabilities are used by the $(AA)^2$ model to determine the most likely number and types of units located at a given surface node or arc. The perception is stored in vector format with each element, j, representing the number of *jth* type units located at the node or arc;

thus, probabilities are computed for each possible combination of units. Readers interested in the actual derivations of these calculations are referred to the Master's Thesis of CPT Karl Schmidt, USA. [Ref. 6] The decisions made by each side in the model are based on the values of these perceptions. This information is used to determine what enemy units are selected as potential targets by the $(AA)^2$ model. For each unit type a determination is made of the most likely number present on a node. For example, assume that three types of units are defined: armor, mechanized, and infantry brigades. The perception vector for a node or arc would be of the format

(# armor brigades, # mechanized brigades, # infantry brigades)

with each element representing the number of brigades of the unit type located on the node. Assume that the perception vectors for a given node are

(1,1,1) = 0.05
(1,2,1) = 0.30
(1,2,2) = 0.25
(2,1,1) = 0.20
(2,1,2) = 0.10
(2,2,2) = 0.10

representing the individual probabilities of different combinations of the unit types on the node. To determine the most likely number of armor brigades on the node, first the probabilities associated with perception vectors containing one armor brigade are added together, (0.05 + 0.30 + 0.25 = 0.60) and compared to the sum of the probabilities for vectors with two armor brigades (0.20 + 0.10 + 0.10 = 0.40). For the example, the probability of there being one armor brigade located at the node is greater than for two armor brigades. This is repeated for each type of unit and results in the most likely combination of unit types and quantities on each node in the surface network. At present this feature is not modeled in the $(AA)^2$ model, but will be included when introduced in the JWAEP model. The $(AA)^2$ model assumes that the unit combination with the highest probability is what is located at the node and uses this in the calculations of the enemy strength at the node. Each perceived unit instance becomes a potential target (POT) and is subject to attrition from air assets. For the $(AA)^2$ model, the perceived values of one side

are used to apportion and allocate air assets for air attacks against the other side. The $(AA)^2$ model assumes that the target data given to it by JWAEP are the most current and accurate data available at the beginning of the air planning cycle. The actual result of air attack sorties depends on the accuracy of this information.

Each side represented in JWAEP has one or more courses of action (COAs) defined for ground and naval forces. These represent the possible composition, routes, and actions of forces within the theater. The COAs for each side are defined based on the expected enemy force composition and movement. Each enemy COA is defined as a collection of units moving along one or more avenues of approach. An avenue of approach is a collection of connected ground nodes and arcs. JWAEP calculates the probability that the enemy is pursuing a specific COA based on sensor detection and the perception of the number and type of units perceived on a ground node. The COA probabilities change as enemy units are detected and their movements are inferred. The $(AA)^2$ model uses the COA probabilities to weight the value of units based on their contribution to the enemy's perceived COA.

IV. METHODOLOGY

In this chapter, methods are formulated to apportion and allocate theater tactical aircraft for air-to-surface attack missions. These missions will be referred to as air interdiction (AI) missions and include attacks on ground and naval units, air bases, logistics units, C^3I units, and air defense units. Most theater models require the user to directly assign apportionment values prior to or during a model run. These values then remain static during the run unless specifically changed by the user. Unfortunately, these values must take into account many factors that may be unknown to the user before the model run, such as what enemy strengths should be targeted by air attacks, what effect air attacks will have on these strengths, and how operations of other types of forces (ground and naval) will affect the enemy strengths. Requiring the user to assign values for the apportionment of air assets in advance prevents the analysis of how changes in a campaign effect the apportionment and allocation of air assets and vice-versa. A more dynamic way of assigning apportionment values is to have the user define what objectives are to be accomplished by air assets, and dynamically apportion and allocate air sorties to support them. This latter approach allows the user to identify the objectives to be accomplished by AI missions during a campaign and what constitutes successful completion of the objectives.

A. STRENGTH CATEGORIES

The first step in determining the apportionment of air assets to AI missions against potential targets is to translate the objectives of the campaign into values that can be used by the $(AA)^2$ model. For AI attacks, an objective calls for the reduction of an enemy capability. This model divides the capabilities of an enemy unit into 20 different strength categories. Each potential target will have values calculated for each strength category dependent on what equipment is assigned to the unit. The types and amounts of equipment attached to a unit may be based on "ground truth" or "perceived" values

depending on the what type of information is available. These categories are grouped into attrition, C3I, mobility, and logistics strengths and are listed with their corresponding 3-character identifying code.

- **Attrition Strengths** The attrition strengths summarize a unit's weapon systems capability to attrite other units into categories relating to the medium in which they are located and the medium of the unit they attrite.
  - Air-to-Air attrition strength (AAA)
  - Air-to-Surface attrition strength (AAS)
  - Air-to-Subsurface (Underwater) attrition strength (AAU)
  - Surface-to-Air attrition strength (ASA)
  - Surface-to-Surface attrition strength (ASS)
  - Surface-to-Subsurface attrition strength (ASU)
  - Subsurface-to-Surface attrition strength (AUS)
  - Subsurface-to-Subsurface attrition strength (AUU)
- **Command, Control, Communication, and Intelligence (C^3I) Strengths** C^3I strengths summarize a unit's C^3I systems capability to process information (first three categories) and the ability to resist enemy efforts to restrict the capabilities.
  - Command and Control strength (CC2)
  - Intelligence strength (CIN)
  - Communication strength (CCO)
  - Counter-C3 strength (CCM)
- **Mobility Strengths** Mobility strengths summarize a unit's mobility systems capability to move or support the movement of equipment and the capability of the units' systems to restrict enemy movement.
  - Mobility strength (MMO)
  - Counter-Mobility strength (MCM)

- **Logistic Strengths** Logistic strengths summarize a unit's logistic systems capabilities to store POL and ammunition, their capability to support the storage, and their ability to transport POL and ammunition via air and surface transport mediums.
    - POL logistic strength (LPO)
    - Ammunition logistic strength (LAM)
    - Air Support logistic strength (LAS)
    - Surface Support logistic strength (LSS)
    - Air Transportation strength (LAT)
    - Surface Transportation strength (LST)

To translate a campaign objective into one or more strength categories, the user first decides which categories are associated with the objective. This decision is made by determining which enemy capabilities are to be targeted by AI attacks and relating them to the strength categories. For example, the first phase of the campaign plan may be for friendly forces to gain air superiority over enemy airspace. The objectives associated with this phase may include:

- Objective 1: Destruction of enemy fighter aircraft on the ground.
- Objective 2: Destruction of enemy surface-to-air defense systems.
- Objective 3: Degradation of the enemy's ability to control air defense systems.
- Objective 4: Destruction of usable runway length at air bases.

It is desired that each of these objectives be stated in terms of the strength categories and the types of units to be attacked. This is done by selecting the strength categories and type of enemy units that relate to the objective to be achieved. For the example, the objectives are related to strength categories and unit types as shown in Table 1. The numbers in the table represent the objectives. Objective 1 relates to reducing the enemy's

air-to-air capability. Since it is assumed that enemy air-to-air capability is limited to air-to-air capable aircraft, fighter and attack squadrons were selected for the AAA strength category. The second objective relates to reducing the enemy's surface-to-air capability. Since surface-to-air defense systems can be attached to any unit type, all unit types were selected for the ASA strength category. The third objective deals with reducing the command and control capability of major components of the enemy's air defense structure. The units that coordinate air defense systems include air bases, C^3I units, ground units and naval units. These units were selected for the CC2 strength category.

	Strength Categories			
Unit Types	**AAA**	**ASA**	**CC2**	**MMO**
Air Bases		2	3	4
Fighter Squadrons	1	2		
Attack Squadrons	1	2		
Ground Units		2	3	
Naval Units		2	3	
Logistic Units		2		
C^3I Units		2	3	

Table 1. Phase objectives by unit type and strength category.

The final objective deals with reducing the amount of usable runway available for enemy air bases. For air bases, the MMO strength represents the capability of the air base to launch aircraft and is selected to account for the fourth objective. The actual selection of strength categories for each phase of a campaign is left to the discretion of a user. The $(AA)^2$ model allows any number of the strength categories to be selected in a campaign phase.

Following the selection of the strength categories, the user must decide what amount of reduction is desired for each of the unit and strength combinations. These are expressed as fractions of the initial strength of all units within the category. Fractions of a unit's total strength are used instead of absolute values to make the transition from phase

objectives to desired levels easier for a user. A user may elect to calculate absolute desired strengths and translate them into desired level fractions. For each unit type and strength category combination, the user defines what amount of reduction is desired to achieve the objective. For the example above, values for each unit type and strength category are illustrated in Table 2. Analysis is possible using the $(AA)^2$ model on the effect of different desired levels on the apportionment and allocation of air assets during a campaign. Inputs as shown in Table 2 are made for each phase of the air campaign, relating the objectives of the phase to enemy capabilities to be reduced by AI attacks.

	Strength Categories			
Unit Types	**AAA**	**ASA**	**CC2**	**MMO**
Air Bases		0.1	.3	.25
Fighter Squadrons	0.1	0.1		
Attack Squadrons	0.5	0.1		
Ground Units		0.1	.3	
Naval Units		0.1	.3	
Logistic Units		0.1		
C^3I Units		0.1	.3	

Table 2. Possible desired fraction of strength for phase objectives.

B. STRENGTH VALUES

1. Data and Variable Definitions

The $(AA)^2$ model calculates the strength of potential targets for each strength category based on data generated by the model and from user input. To begin the process a list of potential targets (POTs) for AI attacks is generated. This list is dependent on what enemy units have been detected by friendly intelligence assets and defines what type of unit the POT is, what equipment it currently has, and where it is located. This list is

updated at the beginning of each air planning period as new targets are added and the description of current targets change. Given the unit type for a POT p, the Table of Organization and Equipment (TO&E) gives the quantity of each equipment type e in the unit represented by the variable TOE_{pe}. Also provided in the POT list is the amount of equipment inferred to be operational in the unit. This inferred quantity is determined by what friendly sensors have detected with the unit, represented by the variable CNE_{pe} for each POT p and equipment type e. Each equipment type has values assigned for each strength category that represent the equipment's strength relative to other equipment types. These values are assigned by the user and remain constant during a model run. Within each strength category these values should represent a relative ranking of the equipment types. These equipment strength values are represented by ESV_{es} for all equipment types e and strength categories s.

The final data needed to calculate strength values are values representing the value of each POT to its side's expected course of action and the organizational level at which it is operating. The COA value is represented by the variable COA_p for each POT p. As was mentioned in the discussion of the JWAEP model, each side represented has one or more possible COAs defined. Each COA has associated with it an avenue of approach that defines the nodes and arcs the enemy COA will travel along. All POTs are assigned a value for COA_p based on the COA probability (from JWAEP) associated with the node (or arc) on which they are located. This value will change as the actual enemy COA becomes apparent to friendly forces. The $(AA)^2$ model updates the COA probabilities at the start of each air planning period. An additional value is assigned for each POT that is used to weight the strength of a unit to represent the level at which the unit is operating. This gives different weights to POTs of the same unit type but of different sizes. It is assumed that units at the operational level should be valued more than those at the tactical level. This especially comes into play when looking at C^3I and logistic units. Destruction of equipment attached to higher echelon units will greatly effect the capabilities of lower echelon units. This value is assigned by the user and represented by the variable ECH_p for

all POTs. This value is a real number greater than 1.0, with greater values given to units operating at higher echelons.

2. Calculation of Strength Values

The first level of the $(AA)^2$ model calculates values used to determine if the stated objectives of a phase have been met. This is done for each unit and strength category combination. The first value, TS_{ps}, represents the strength of a full strength unit, defined with respect to the unit's TO&E (used to define the unit in the model). The second value, CS_{ps}, represents the strength computed using the current quantities of equipment and weapon systems attached to the enemy unit as perceived by friendly forces. The third value, DS_{ps}, represents the strength desired by the user at completion of an associated objective; stated as the fraction of TS_{ps}. These variables are summarized as follows:

- TS_{ps}: TO&E strength of POT p for strength category s.
- CS_{ps}: Current strength of POT p for strength category s.
- DS_{ps}: Desired strength of POT p for strength category s.

These values are calculated at the beginning of each air planning cycle and are used to determine whether a POT's strength in each category has been reduced to the desired level. If CS_{ps} is less than or equal to DS_{ps} then the POT's desired strength in the category has been achieved. If CS_{ps} is greater than DS_{ps} then air attacks are needed to reduce the POT's strength in the category to the desired level. The total and current strengths of a POT p in strength category s are calculated by Equations (1) and (2).

$$TS_{ps} = \left(1 + COA_p\right) \times ECH_p \times \sum_e \left(TOE_{pe} \times ESV_{es}\right) \quad (1)$$

$$CS_{ps} = \left(1 + COA_p\right) \times ECH_p \times \sum_e \left(CNE_{pe} \times ESV_{es}\right) \quad (2)$$

The value of COA_p is a probability and is therefore on the scale [0,1]. To ensure that enemy units that are not associated with a defined COA are assigned a non-zero value, 1.0 is added to COA_p. The desired strength of a POT is calculated by Equation (3).

$$DS_{ps} = DL_s\left(UnitType(p)\right) \times TS_{ps} \quad (3)$$

where,

$$UnitType(p) = \text{ Unit Type of POT } p.$$

The amount of reduction in strength, DRS_{ps}, needed for each strength category of each POT is calculated using Equation (4).

$$DRS_{ps} = \boldsymbol{MAX}\left[0, CS_{ps} - DS_{ps}\right] \quad (4)$$

These values are used to allocate AI sorties during the air allocation portion of the $(AA)^2$ model.

C. ALLOCATION

The second level of the $(AA)^2$ model determines the optimal allocation of AI sorties to POTs. The Mixed Integer Program (MIP) used in this allocation was based on work done by Major Brian Griggs, USAF in a Master's Thesis. [Ref. 7] The objective of the MIP is to maximize the total reduction in strength for all strength categories while minimizing the amount of expected attrition of friendly air assets. Constraints are used to limit the number of sorties flown, ensure the target is within range of an aircraft and to prevent destruction of a POT below the desired strength. The allocation is for the entire air planning period (normally 24 hours) and gives as output flight packages consisting of the number of aircraft in the group, the POT they are to attack, what equipment types to

target and whether fighter or electronic warfare (SEAD) escort aircraft are to be provided. It is assumed that the optimal path to a POT and the expected attrition on that path are know before the allocation of assets is determined. The following sections define the data used by the allocation MIP and how they are obtained.

1. MIP Variable Definitions

The following variables are used in the air allocation MIP definition:

- b: Air base identification number.
- a: Aircraft type.
- p: Potential target identification number.
- s: Strength category identification number.
- e: Equipment type identification number.
- f: Escort fighter protection (with, without).
- j: Escort SEAD protection (with, without).
- $SORTIES_{bapsefj}$: Decision variable representing the number of AI sorties of a flight group to fly from air base, b, by aircraft, a, assigned to POT, p, reducing strength, s, destroying equipment, e, with (or without) fighter escort, f, and SEAD escort, j.
- $FIGHT_{bap}$: Integer decision variable representing the number of fighter escort packages assigned to escort flight group made up of aircraft, a, flying from air base, b, against POT, p.
- $SEAD_{bap}$: Integer decision variable representing the number of electronic escort (SEAD) packages assigned to escort flight group made up of aircraft, a, flying from air base, b, against POT, p.
- TSD_{ps}: Calculated target strength destroyed by flight group for POT, p, strength, s.
- $EA_{bapsefj}$: Calculated expected attrition suffered by flight group consisting of aircraft, a, flying from air base, b, against POT, p, reducing strength, s, attacking equipment, e, with (or without) fighter escort, f, and SEAD escort, j.

- PK_{ae}: Probability of kill by one sortie of aircraft, a, against equipment, e.
- ESV_{es}: Strength value of equipment, e, for strength, s.
- DRS_{ps}: Desired reduction for POT, p, in strength, s.
- CNE_{pe}: For POT, p, the current quantity of equipment, e.
- $MAXSORTIES_{ba}$: Maximum sorties available by aircraft, a, from air base, b.
- $MAXFIGHT_b$: Maximum escort fighter packages available from air base, b.
- $MAXSEAD_b$: Maximum escort SEAD packages available from air base, b.
- $MAXFTRESC$: Maximum number of AI sorties that can be escorted by one fighter escort package.
- $MAXSEADESC$: Maximum number of AI sorties that can be escorted by one SEAD escort package.
- $RANGE_{bap}$: Matrix with an entry of 1 if aircraft, a, from air base, b, can reach POT, p, along optimal route path.
- $PSAA_{bapf}$: Probability of survival due to enemy air-to-air assets for an aircraft, a, from air base, b, attacking POT, p, with (or without) attached fighter escort, f, along optimal route path.
- $PSSA_{bapj}$: Probability of survival due to enemy surface-to-assets for an aircraft, a, from air base, b, attacking POT, p, with (or without) attached SEAD escort, j, along optimal route path.
- COA_p: Course of action probability for POT, p.
- ECH_p: Echelon weight for POT, p.
- W: user defined maximum attrition.

2. MIP Formulation

The air allocation program formulation is as follows:

Objective function.

$$\textbf{\textit{MAXIMIZE}} \sum_{p}\sum_{s} TSD_{ps} \tag{5}$$

Subject to:

Calculation of target strength destroyed by AI sorties

$$TSD_{ps} = \left(1 + COA_{p}\right) \times ECH_{p} \times \sum_{b}\sum_{a}\sum_{e} RANGE_{bap} \times PK_{ae} \times ESV_{es} \times \sum_{f}\sum_{j}\left(SORTIES_{bapsefj} - EA_{bapsefj}\right) \tag{6}$$

Calculation of expected attrition suffered by AI sorties.

$$EA_{bapsefj} = SORTIES_{bapsefj} \times \left(1 - PSAA_{bapf} \times PSSA_{bapj}\right) \tag{7}$$

Limit the attrition suffered by a flight group to no more than user defined.

$$\sum_{s}\sum_{e}\sum_{f}\sum_{j}\left(EA_{bapsefj} - W \times SORTIES_{bapsefj}\right) \le 0 \tag{8}$$

Limit AI sorties assigned to those available.

$$\sum_{p}\sum_{s}\sum_{e}\sum_{f}\sum_{j} SORTIES_{bapsefj} \le MAXSORTIES_{ba} \tag{9}$$

Limit equipment kills to no more than the current quantity.

$$PK_{ae} \times \sum_{f}\sum_{j}\left(SORTIES_{bapsefj} - EA_{bapsefj}\right) \le CNE_{pe} \tag{10}$$

Limit target strength destroyed to no more than are desired.

$$TSD_{ps} \le DRS_{ps} \tag{11}$$

If fighter escort is allocated, require one fighter escort package for every MAXFTRESC AI sorties protected.

$$\sum_{s}\sum_{e}\sum_{j} SORTIES_{bapselj} \leq FIGHT_{bap} \times MAXFTRESC \tag{12}$$

If SEAD escort is allocated, require one SEAD escort package for every MAXSEADESC AI sorties protected.

$$\sum_{s}\sum_{e}\sum_{p} SORTIES_{bapsef1} \leq SEAD_{bap} \times MAXSEADESC \tag{13}$$

Limit the number of fighter escort packages used to no more than available.

$$\sum_{a}\sum_{p} FIGHT_{bap} \leq MAXFIGHT_{b} \tag{14}$$

Limit the number of SEAD escort packages used to no more than available.

$$\sum_{a}\sum_{p} SEAD_{bap} \leq MAXSEAD_{b} \tag{15}$$

Non-negativity constraint for number of AI sorties.

$$SORTIES_{bapsefj} \geq 0 \tag{16}$$

Non-negativity constraint for amount of expected attrition.

$$EA_{bapsefj} \geq 0 \tag{17}$$

3. Allocation Program Output

The output of the air allocation LP is similar to an Air Tasking Order (ATO) used by theater commanders to allocate aircraft to sorties, missions, and targets. The allocation program builds flight groups made up of AI sorties and any fighter and SEAD escort packages. These flight groups are scheduled to fly from a friendly air base to an enemy unit and attack a specified equipment type. These data are used by the $(AA)^2$ model's target attrition methods. The number of sorties assigned for each aircraft type, air base, equipment type, and target combination is for the entire air planning period.

D. PHASE ACTIVATION AND TRANSITION

At the beginning of model execution, only the first phase is considered to be active. Phases are sequentially activated based on a determination of whether the strength categories of the previous phase have been sufficiently reduced. If it is determined that the next phase should be activated, its computed strength values are used to allocate sorties to targets. Transition from one phase to one or more following phases occurs as fewer sorties are allocated to one phase and the remaining available sorties are allocated to follow-on phases. The $(AA)^2$ model makes the decision to activate follow-on phases based on whether the strength values of the current phase have been reduced to within a user specified fraction of the desired values. Once the strength values are within the specified fraction, any sorties not allocated to the current phase are allocated to the next phase. An assumption made is that the phases specified by the user are in order of their importance. Therefore, sorties will be allocated to the first phase, then any remaining will be allocated to the next active phase, and so on. Phases are not de-activated during the model run. This accounts for the possibility that enemy strengths that have previously been reduced to the desired level may increase above the desired level due to reinforcements or repair of equipment.

The calculations used to determine if the next phase should be activated are described below. The criteria used ensure that the total strengths over all POTs have been reduced to within the user defined range of the desired strength levels. Before each model run the user defines the value, *CriteriaVal*, used to determine a range from the phase's desired strength values at which the next phase is activated. For each active phase i the desired reduction in strength fraction, $DRSF_s$, is calculated using Equation (18).

$$DRSF_S = \sum_p \frac{DRS_{ps}}{\left(TS_{ps} - DS_{ps}\right)} \tag{18}$$

The next phase will be activated if the value of $DRSF_s$ for all strength categories s of the phase are less than or equal to *CriteriaVal*. Any available sorties remaining following

allocation to the phase will be allocated to the next phase. This is calculated for each active phase in the campaign.

The selection of *CriteriaVal* is dependent on how quickly a user wants successive phases to be activated. A large value for *CriteriaVal* (> 0.5) will result in phases being activated while preceding phases are still far from their desired levels. This situation closely resembles phases being executed concurrently, since many phases may be active at the same time. Setting the value of *CriteriaVal* to a low number (< 0.2) will result in phases being executed sequentially. The activation of a phase will occur only when the preceding phase has been reduced close to its desired levels.

V. MODEL DESCRIPTION AND DEMONSTRATION

This chapter discusses the specifics of the implementation of the $(AA)^2$ model and demonstrates its use in several examples. Two software applications were used in the implementation of the model. Borland Pascal Version 7.0 was used to implement the air apportionment and air attrition simulation portion of the $(AA)^2$ model. This portion includes procedures for creating the input files used by the sortie allocation MIP. The sortie allocation MIP was implemented using GAMS Version 2.03. These applications were selected because they were both capable of being implemented on a personal computer and the author's familiarity with them. The examples used in this chapter are based on a relatively simple air campaign that consists of two phases. These examples are sufficient to demonstrate the methods used in the model and the type of analysis that can be conducted.

A. MODEL OVERVIEW

The $(AA)^2$ model consists of a main Pascal program and a sortie allocation MIP. The Pascal program is used to calculate the strength values associated with each POT, build the enemy air defense coverage for use in determining the optimal routes from friendly air bases to POTs, and simulate the attrition of POTs by sorties generated by the allocation MIP. Data files used by the sortie allocation MIP are generated by the Pascal program. The output of the sortie allocation MIP is read into the Pascal program to begin the process for the next day of the campaign. Figure 5 shows the general process that occurs for each day of the simulation. Appendix B contains the Pascal source code. Appendix C contains a sample listing of the GAMS air allocation MIP used by the $(AA)^2$ model.

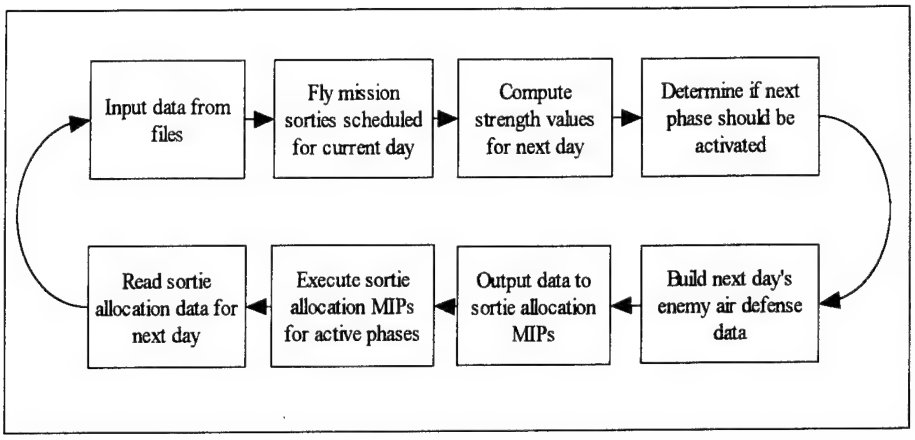

Figure 5. Model Flow.

1. Data Structure

The data variables used in the $(AA)^2$ model and their structure are found in Appendix A. All data used by the Pascal program and the sortie allocation MIP are stored in text files. These files preserve data for use during the next day as well as archiving data for model analysis. For each day of the campaign, a synopsis of the Pascal program's execution is archived to a "results" text file for post-analysis. The files created by the Pascal program are also listed in Appendix A along with their descriptions.

2. Attrition

After input and initialization of data, the $(AA)^2$ model simulates the air interdiction (AI) and fighter/SEAD escort missions generated by the sortie allocation MIP. A mission is defined as a grouping of a specified number of AI sorties, fighter and SEAD escort packages. The AI sorties of the mission will attack a specific type of equipment attached to a target unit. The SEAD escort packages (if attached) will attack any surface-to-air missile systems attached to the target unit. The fighter escort packages (if attached) do not currently attack any enemy fighter aircraft since the model does not explicitly simulate airborne enemy aircraft. Target attrition by the SEAD escort packages is conducted first.

SEAD escort packages are not subject to attrition by enemy air defenses, as it is assumed that SEAD escort aircraft will normally stand-off from enemy air defenses while supporting AI mission sorties. SEAD escort packages attack enemy surface-to-air missile (SAM) systems attached to a target unit (if any exist). The number of SAMs killed, *SAMKilled*, by the SEAD package is determined by drawing a random sample from a binomial distribution. The parameters of the distribution are the number of SEAD package shots, *SEADShots*, and the probability of kill, $PKSEAD_e$, for the enemy SAM system type e. The value drawn from the binomial distribution is assigned to the value *SAMKilled* and equals the number of shots by a SEAD package that hit enemy SAM systems. This number is subtracted from the number of SAM systems attached to the target. If *SAMKilled* is greater than the current number of enemy SAM systems attached to the target, the number of systems is set to zero.

The AI missions of each flight group are flown next. The AI mission sorties are first subject to enemy air-to-air and surface-to-air attrition. Again, the number of AI sorties killed by enemy defenses is determined by drawing a random sample from a binomial distribution. The probability of kill of an AI mission sortie flying from an air base to a target unit is derived from the $PSAA_{bapf}$ and $PSSA_{bapj}$ values used in the allocation MIP. The probability of kill, $PKill_{bap}$, of each AI mission sortie is calculated using Equation 19.

$$PKill_{bap} = 1 - \left(PSAA_{bapf} \times PSSA_{bapj} \right) \tag{19}$$

The parameters used for the binomial draw are the number of AI sorties flying to a target, $SORTIES_{bapseffj}$, and the probability of kill due to enemy air defenses, $PKill_{bap}$. The random sample drawn is assigned to the variable *ACKilled*. This value is subtracted from $SORTIES_{bapseffj}$ to calculate the number of AI sorties remaining to attack the target unit. The number of equipment systems killed by the AI sorties is again calculated by drawing from a binomial distribution with the parameters set to the number of remaining AI sorties and the probability of kill, $PKAS_{ae}$, of equipment type e by aircraft type a. The value drawn is number of equipment systems killed, *EquipKilled*, by the AI sorties. The value,

EquipKilled, is subtracted from the current number of equipment, CNE_{pe} of type, e, attached to POT, p. If *EquipKilled* is greater than CNE_{pe}, then CNE_{pe} is set to zero. Appendix D contains listings of the probabilities of kill for enemy defense systems vs. friendly aircraft and friendly aircraft vs. equipment types.

3. Strength Computation and Phase Activation

After all SEAD and AI missions have been flown, the strengths of each POT are recalculated. The values of TS_{ps}, CS_{ps}, DS_{ps}, and DRS_{ps} are calculated for each POT and strength category. These data are formatted and outputted to files for use by the allocation MIP. These values are also used to determine whether the current phase's strengths have been reduced to below the user specified value, *CriteriaVal*. If so, the next phase (if there is one) is activated.

4. Enemy Air Defense System

The enemy air defense system is built using the Pascal program developed by LT Wang, R.O.C. Navy, in a Master's Thesis. [Ref. 8] The algorithms developed in his thesis have been implemented in JWAEP and are consistent with the methods used in the $(AA)^2$ model. For each enemy SAM system or air-to-air capable aircraft a maximum range and a probability of kill, PK, against friendly aircraft type are defined. This information, along with the location of the system, is used to calculate the PK by enemy air-to-air and SAM systems in each air grid for each type of friendly aircraft. These values are then used to determine the optimal route from the friendly air base to each POT. This optimal route gives an expected PK for enemy air-to-air and surface-to-air threat systems and a range from the friendly air base to the POT. Each type of friendly aircraft has a defined maximum range. If the range to a POT along the optimal route is farther than an aircraft's maximum range, the variable $Range_{bat}$ is assigned a value of zero. Otherwise, it is given a value of one. The expected PK of enemy air-to-air and surface-to-air threat systems along the optimal route are use to compute the probabilities of success for each friendly aircraft

type flying from an air base to a POT. Two values are calculated for the probability of success for both enemy air-to-air (PSAA) and surface-to-air (PSSA) threats for an aircraft type, a, flying from an air base, b, to a POT, p. The first value assumes no escort fighter escort, f, or SEAD escort, j, package is attached. These values, $PSAA_{bapf}$ and $PSSA_{bapj}$, are equal to one minus the respective probabilities of kill calculated for the optimal route. It is assumed that if fighter or SEAD escort aircraft are attached, the respective probability of kill along the optimal route is reduced by a fractional amount, *PKReduction,* assigned by the user. This is consistent with the assumption that allocation of escort aircraft to a flight group should greatly reduce the threat to AI sorties. The resultant PK values are then used to calculate the values of $PSAA_{bapf}$ and $PSSA_{bapj}$ with attached escort packages. These data values are formatted and outputted for use by the allocation MIP.

5. Allocation of Sorties

AI, escort fighter and SEAD mission sorties are allocated by the allocation program following completion of the Pascal program. Sorties are allocated to Phase One first, then if any sorties remain, are allocated to Phase Two, Three, etc. (if active). The output of the allocation program is transferred to mission input files that are read for the beginning of the next day of the campaign by the Pascal program. To ensure a fast allocation solve time, the variable for the number of AI sorties is relaxed from an integer variable to a real variable. Since this will result in fractional AI sorties being allocated, only the integer part of the variable is transferred to the mission input files. While this does not result in an "optimal" solution, it is considered to be a "near" optimal solution sufficient for the execution of the $(AA)^2$ model. Test runs of the model using the "near" optimal solution compared closely to runs requiring integer solutions for AI sorties.

B. MODEL DEMONSTRATION

A simple scenario was constructed to demonstrate the use of the $(AA)^2$ model. Figure 6 gives a graphical view of the battle area. The ground node network consists of

18 nodes. Two enemy COAs were defined for the scenario. Each COA consist of an equal sized force with different avenues of approach (AOA). The AOAs for COA 1 and COA 2 are associated with nodes 1-7 and 8-14, respectively. At the beginning of the scenario the COAs are equally likely. As the scenario progresses, the COA probabilities are changed with COA 2 becoming the most likely. The COA probabilities of each COA for each day of the campaign are listed in Table 3. Eight POTs are defined and listed in Table 4. The units moving along COA 1's AOA include Armor Brigade 1, Artillery Battalion 1, and Logistic Unit 1. Armor Brigade 2, Artillery Battalion 2, and Logistic Unit 2 proceed along the AOA for COA 2. The enemy air base and logistic base remain stationary throughout the scenario. The movement and node locations of the POTs for each day of the campaign are listed in Table 5.

Day	COA 1	COA 2
1	0.5	0.5
2	0.4	0.6
3	0.3	0.7
4	0.2	0.8
5	0.2	0.8
6	0.2	0.8
7	0.2	0.8
8	0.2	0.8

Table 3. Course of Action (COA) probabilities.

Each POT has a TO&E defined which gives the initial quantities of the 10 equipment types defined in Table 6. These are standard equipment types taken from the THUNDER model data descriptions. [Ref. 4] The TO&E data and equipment strength values for each type of equipment are contained in Appendix D. Two friendly air bases are defined with each having two types of friendly aircraft. Air base 1 (node 17) represents a fixed air base which has attached 20 F-117A's and 30 F-15E's. Air base 2 (node 18) represents a aircraft carrier that has attached a 20 A-6Es and 30 F-18Cs. Each air base has 12 fighter and SEAD packages available per day.

Air Base
Armor Brigade 1
Armor Brigade 2
Artillery Battalion 1
Artillery Battalion 2
Logistic Unit 1
Logistic Unit 2
Logistic Base

Table 4. Potential targets.

	Day							
Potential Target	1	2	3	4	5	6	7	8
Air Base	15	15	15	15	15	15	15	15
Armor Brigade 1	2	3	4	4	4	4	4	4
Armor Brigade 2	9	10	11	12	13	14	14	14
Artillery Battalion 1	1	2	3	3	3	3	3	3
Artillery Battalion 2	8	9	10	11	12	13	13	13
Logistic Unit 1	1	2	3	3	3	3	3	3
Logistic Unit 2	8	9	10	11	12	12	12	12
Logistic Base	16	16	16	16	16	16	16	16

Table 5. Potential Target (POT) node locations by campaign day.

Tank
Artillery
Attack Jet
Fighter Jet
Fixed SAM
Mobile SAM
C^2 Van
Runway Section
POL Blivet
POL Truck

Table 6. Equipment types.

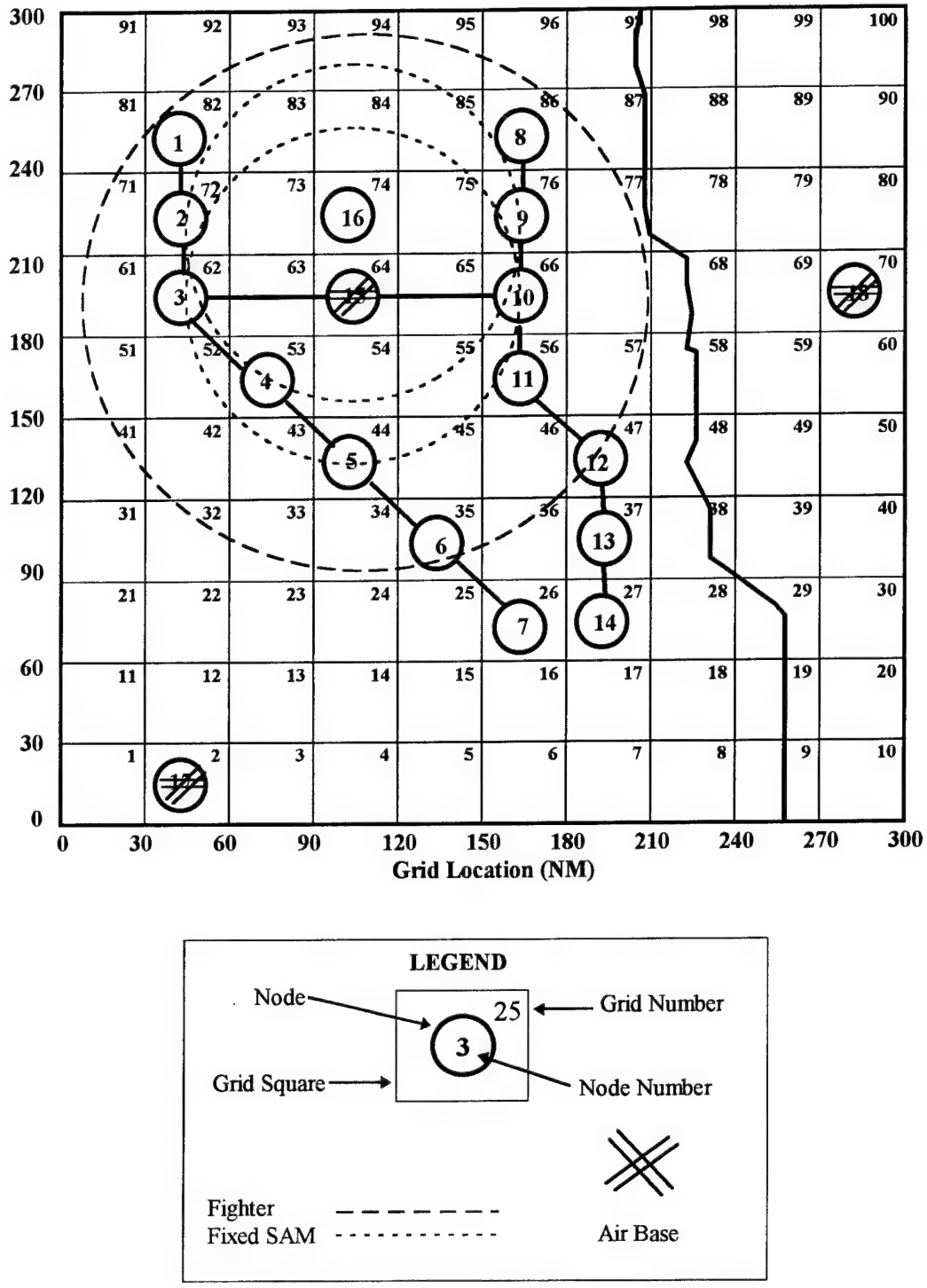

Figure 6. Air grid and node layout.

Phase One of the campaign has as its objective to gain air superiority over the enemy air-to-air and surface-to-air threat. Phase Two's objective is to reduce the enemy's ground combat and logistic supply capability. The strengths and unit types to be targeted for Phase One and Phase Two are given in Table 7 and 8, respectively.

Unit Type	AAA	ASA	CC2	MMO
Air Base	X	X	X	X
Armor Brigade		X	X	
Artillery		X	X	
Logistic Unit		X		
Logistic Base		X		

Table 7. Selected strength categories for Phase One.

Unit Type	AAS	ASS	LPO	LST
Air Base	X		X	X
Armor Brigade		X		
Artillery		X		
Logistic Unit			X	X
Logistic Base			X	X

Table 8. Selected strength categories for Phase Two.

For Phase One, the desired levels for enemy air-to-air (AAA) and surface-to-air (ASA) strengths were set to 0.1. These values were selected to ensure that the majority of enemy threat systems were destroyed before execution of Phase Two objectives. In addition, the command and control, CC2, elements of units controlling threat systems were assigned a desired level of 0.3. Finally, the air base mobility, MMO, strength was set at 0.2. The Phase Two desired levels were determined in much the same way as Phase One. The air-to-surface (AAS) strength was set to 0.1. Surface-to-surface (ASS), logistic POL (LPO), and logistic surface transport (LST) strengths were set to 0.3. These values were selected to ensure significant damage to enemy ground and logistic units. Tables 9 and 10 show the desired levels for Phase One and Phase Two, respectively.

Unit Type	AAA	ASA	CC2	MMO
Air Base	0.1	0.1	0.3	0.2
Armor Brigade		0.1	0.3	
Artillery		0.1	0.3	
Logistic Unit		0.1		
Logistic Base		0.1		

Table 9. Desired levels for Phase One.

Unit Type	AAS	ASS	LPO	LST
Air Base	0.1		0.3	0.3
Armor Brigade		0.3	0.3	
Artillery		0.3	0.3	
Logistic Unit			0.3	0.3
Logistic Base			0.3	0.3

Table 10. Desired levels for Phase Two.

1. Example One

Example One uses a *CriteriaVal* of 0.3. This value was selected to ensure that the strength values associated with Phase One would be close to their desired levels before activating Phase Two. Figure 7 shows the desired reduction in strength fraction remaining at the beginning of each day of the campaign. It can be seen that by the beginning of Day 4, the desired reduction in strength fractions are all below the *CriteriaVal* of 0.3. AI sorties flown on Day 1, 2, and 3 resulted in all DRS fractions reduced below the *CriteriaVal*. Beginning on Day 4, both Phase One and Phase Two were active and sorties were allocated to both. Day 4 is the transition day for the two phases. The majority of sorties went to reducing the strengths associated with Phase Two with the smaller amount going to Phase One. The number of sorties flown is shown in Figure 8. The DRS fractions for Phase Two are reduced below the CriteriaVal on Day 6. If the example had included a third phase it would have been activated and allocated sorties on Day 7. The

relatively low value selected for *CriteriaVal* ensured that Phase One strengths were close to their desired levels before allocating sorties to Phase Two.

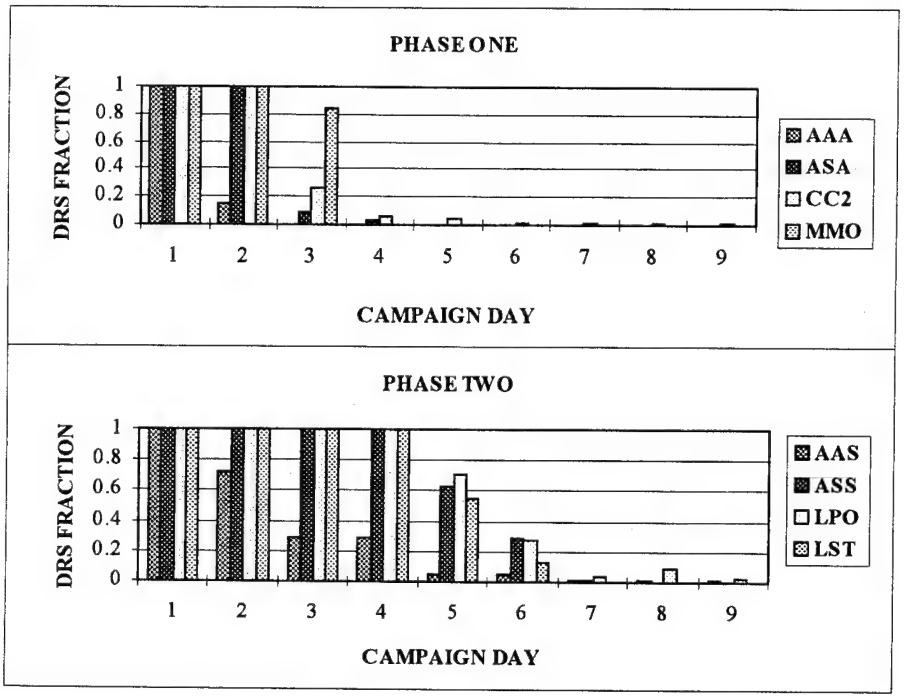

Figure 7. Remaining reduction in strength for Phases One and Two.

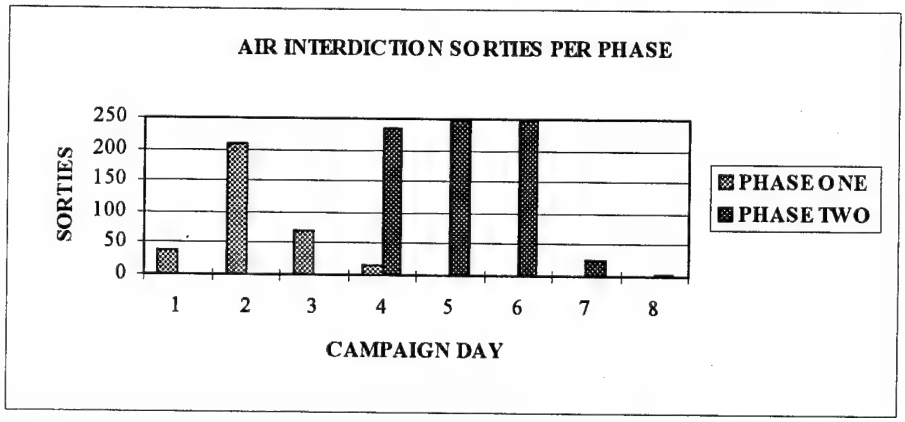

Figure 8. Air Interdiction sorties by campaign phase.

2. Example Two

In Example Two *CriteriaVal* was increased to 0.8. The effect of increasing *CriteriaVal* is to allow activation of follow-on phases while preceding phases are still far from their desired levels. Figure 9 shows the DRS fractions for both phases and Figure 10 shows the sorties flown for each phase. By the start of Day 3, the Phase One strengths had been reduced below the *CriteriaVal* and Phase Two was activated. The increase in the CriteriaVal to 0.8 resulted in Phase Two being activated one day earlier than in Example One. Figure 10 shows the distribution of sorties between the phases. The majority of sorties flown on Day 3 were flown in support of Phase Two objectives. When compared to the first example, the earlier activation of Phase Two allowed excess sorties not used for Phase One objectives to be allocated to Phase Two objectives. This resulted in Phase Two's strengths being reduced earlier than in Example One. This example demonstrates that as the value of CriteriaVal is increased, follow-on phases are activated sooner. This allows for sorties to be allocated to several phases in the same day.

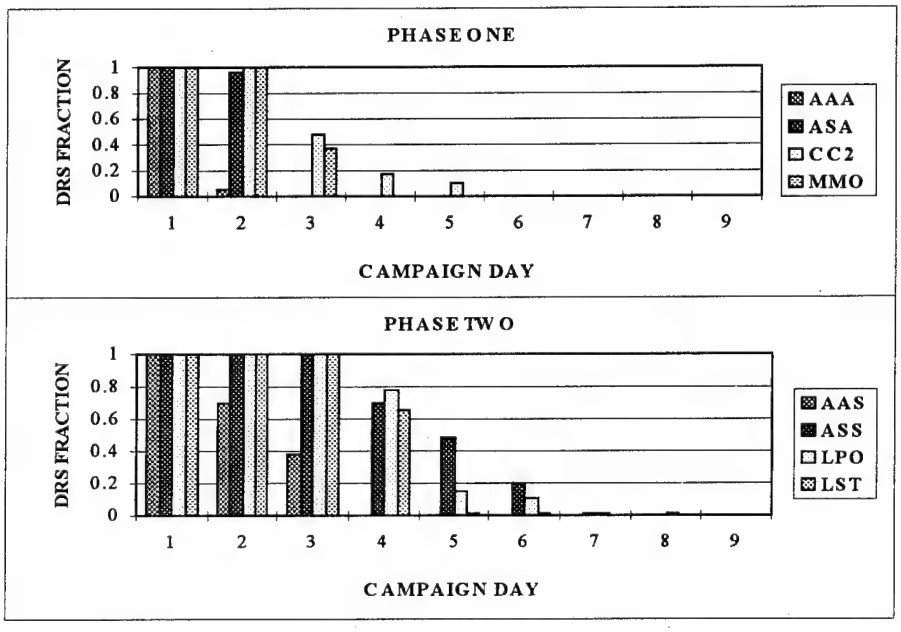

Figure 9. Remaining reduction in strength for Phase One and Phase Two.

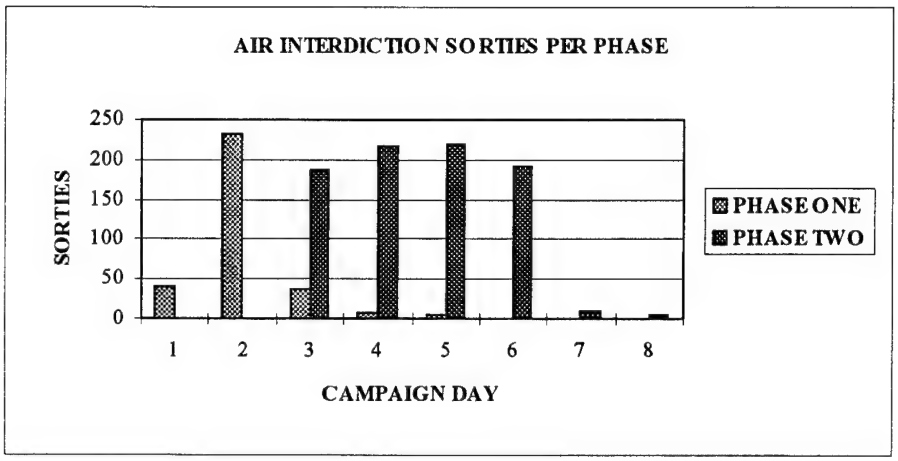

Figure 10. Air Interdiction sorties by phase.

3. Example Three

The purpose of Example Three is to show the effect of an increase in the strength values of an active phase due to enemy reinforcements, updated perceptions, or repair of equipment. This example used the same *CriteriaVal*, 0.3, as Example One. The campaign was conducted much the same way as Example One with Phase Two being activated on Day 4. At the end of Day 5 a reinforcement of fighter aircraft was transferred to the enemy air base and the runways were repaired to full capability. This resulted in an increase of Phase One strengths. In addition, the number of sorties allocated to the Phase One strengths increased for Day 6 with a representative decrease in Phase Two sorties. Figures 11 and 12 show the reduction in strength and sorties flown for each day of the campaign, respectively. One assumption of the $(AA)^2$ model is that sorties are allocated to phases in order of sequence. Therefore, an increase in the strength values of an early phase will result in more sorties being allocated to that phase and less remaining for succeeding phases.

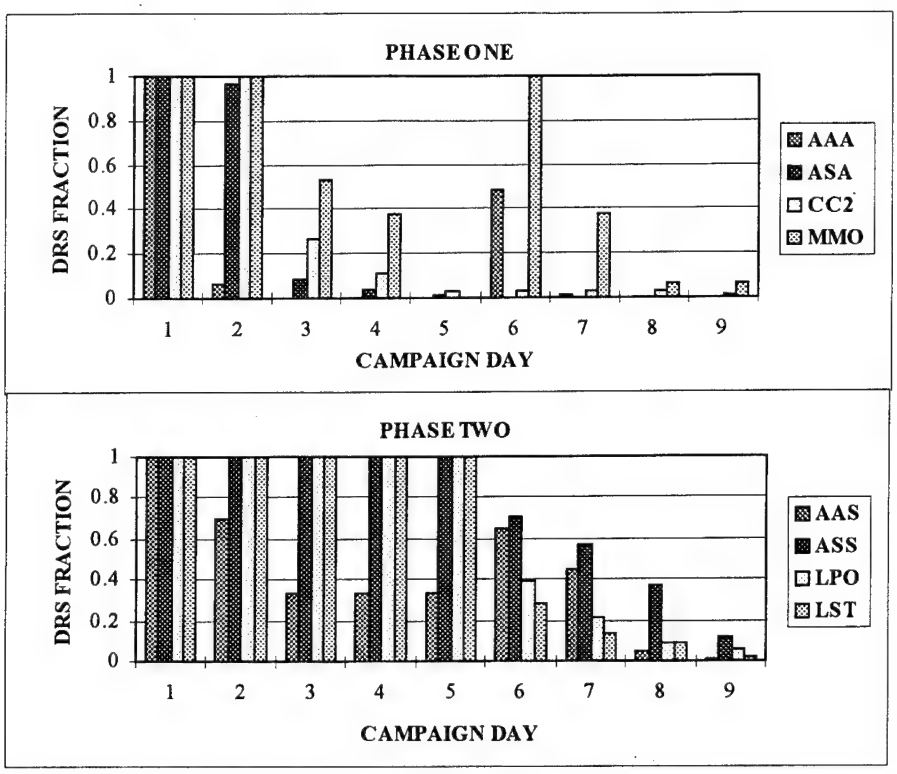

Figure 11. Remaining reduction in strength for Phase One and Phase Two.

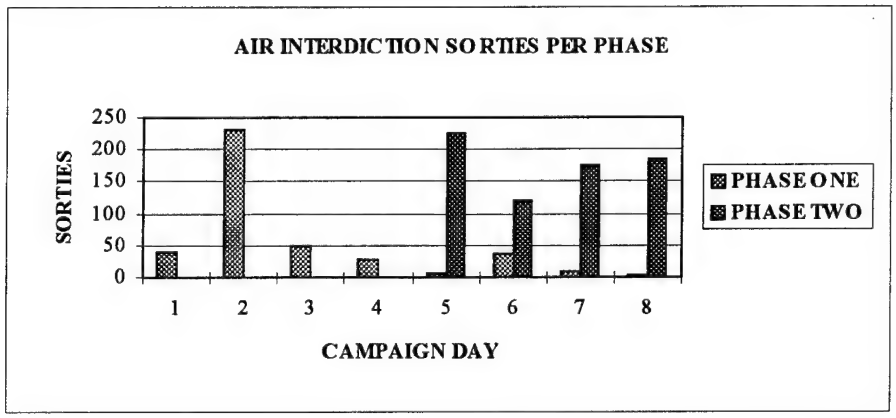

Figure 12. Air Interdiction sorties by campaign phase.

4. Summary

The previous examples demonstrate the $(AA)^2$ model's ability to apportion and allocate air assets based on the objectives of a multiple phase air campaign. In the examples, air assets were initially used to reduce the enemy air defense threat before moving to the objectives of Phase Two. The activation of Phase Two in all the examples was determined by the model based on the current strengths of the POTs and the user selected value of CriteriaVal. The desired levels for Phase One were sufficient to ensure that the enemy air defense threat was reduced to a level that allowed the follow-on phase to be activated and completed. Using the $(AA)^2$ model, analysis can be conducted to determine how changes in desired levels and phase transition criteria affect the apportionment and allocation of air assets.

VI. CONCLUSIONS AND FUTURE RESEARCH

The $(AA)^2$ model provides a process for air campaign planners to analyze the effects of campaign phasing on the apportionment and allocation of theater air assets. In many campaign scenarios, the number of available aircraft sorties is smaller than the number of possible targets. Air planners must make difficult decisions on what targets should be attacked by air sorties on a specified day and which must wait until a later date. The decision of which targets to attack is based on the objectives of the active phase of the campaign. The $(AA)^2$ model gives a method by which not only the order of phases can be analyzed, but also when to activate successive phases. Analysis of the desired level of destruction on different target types is also possible. The methods of the $(AA)^2$ model are also suitable for inclusion into existing theater-level combat simulations. Use of the model in combat simulations removes the need for a user to explicitly define target priorities and schedule air missions.

A. ASSUMPTIONS

Several assumptions made during the development of the $(AA)^2$ model are important to note because they affect the outcome of any analysis conducted using the model. As a stand-alone decision model, the $(AA)^2$ model has several limitations that may affect results. The model deals specifically with air-to-ground attrition. Attrition by other types of units (ground and naval) is not taken into account. At present, the uncertainty regarding what enemy units are located on a ground node is also not considered. This uncertainty is treated in depth in the JWAEP model. The $(AA)^2$ does not attrit fighter or SEAD sorties. The number of fighter and SEAD packages available each day remains constant. The model does not differentiate between the day and night capabilities of different types of aircraft. No determination is made by the allocation program of whether it is an advantage for an aircraft to fly at night and be subject to less air defense threat or fly during the day to increase the probability of destroying a target. While these

assumptions may limit the extent to which the stand-alone version of the $(AA)^2$ model may be used for analysis, the methodology provides the basis for future research.

B. FUTURE RESEARCH

Several areas within the topic of theater air asset apportionment and allocation lend themselves to future research. First, the $(AA)^2$ model deals only with air interdiction (AI) missions and fighter and SEAD escort missions. An improved model would also apportion and allocate other types of missions to include reconnaissance, close air support (CAS), logistic transport, combat air patrol (CAP), and air refueling. To accommodate these mission types, more strength categories will need to be defined which give strengths to related enemy and friendly capabilities. The decision of whether to allocate fighters to protect AI sorties or to fly Combat Air Patrol (CAP) sorties over friendly units is an important area worthy of consideration.

The strength values assigned to each type of equipment represented in the $(AA)^2$ model are assumed to be constant throughout a model run. In reality, the value of a type of equipment for a specific strength category is dependent on several factors that may change during the execution of the model. For example, the surface-to-surface attrition strength (ASS) of a tank may be one value if used in the offense and another if used in the defense. The strength value may also be different if the unit to which it is attached is at full strength or is at partial strength. A unit's strength may also be dependent on the amount of logistic support it can receive from its parent unit. Future work on representing these dependencies and ways of implementing dynamic strength values within a model run is of interest.

Implementation of the $(AA)^2$ model methods into the JWAEP model requires that the computer run time for each campaign day be kept to a minimum. A significant amount of the execution time of the $(AA)^2$ model is due to the use of a MIP to allocation air assets. Replacement of the MIP by a fast execution heuristic program could reduce the

execution time significantly. While the solution may be less optimal than the air allocation MIP, the savings in execution time is desirable.

Finally, the COA probabilities used by the model to weight the strength values of units are currently based only on ground COAs. COAs for air and naval forces are also needed. Naval COA probabilities, suitable for use by the $(AA)^2$ model, were discussed by LT Michael B. Fulkerson, USN in his Master's Thesis. [Ref. 9] The formulation of COAs for air forces have yet to be developed. These values may possibly be based on the air apportionment values of a side and the types of missions flown over a given period of time. Inclusion of these COAs into the $(AA)^2$ methods will more accurately model the strength values of these types of units when compared to ground units.

The $(AA)^2$ model is an initial step in the development of decision algorithms to accurately model the apportionment and allocation of air assets in theater campaigns simulations. By removing the need for the user to explicitly define how air assets are apportioned and allocated, significant analysis may be conducted on apportionment and allocation values. This analysis is important to both campaign planners as well as the forces executing campaign plans.

LIST OF REFERENCES

1. Joint Publication, *Joint Pub 3-0: Doctrine for Joint Operations*, September 1993.
2. Horner, Charles A., LTG, "The Air Campaign," *Military Review*, September 1991.
3. *TACWAR Air Analyst Guide*, April 1992.
4. *THUNDER Analysts Manual*, Version 6.1, May 1994.
5. Youngren, Mark A., *The Joint Warfare Analysis Experimental Prototype (JWAEP) User Documentation (Draft)*, Unpublished, November 1994.
6. Schmidt, Karl M., *Design Methodology for FTLM*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1993.
7. Griggs, Brian J., *An Air Mission Planning Algorithm for a Theater Level Combat Model*, Master's Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, March 1994.
8. Wang, Hua-chung, *Development and Implementation of Air Module Algorithms for the Future Theater Level Model*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1994.
9. Fulkerson, Michael B., Jr., *Integration of Naval Forces into the Early Entry Theater Level Model*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1994.

APPENDIX A. LIST OF VARIABLES

This appendix presents a list of the variables that are used in the $(AA)^2$ model Pascal source code. Descriptions of each variable are provided. Index and local variables are not included.

A. PASCAL SOURCE CODE VARIABLES

UNIT IDENTIFIERS

Main	Main module program.
BuildEAD	Module - Generation of enemy air defense densities and data output.
Data	Module - Variable, constant and type definitions.
FlyMsn	Module - Air sortie and target attrition module.
In_Out	Module - Data file input and output module.
Strength	Module - Target strength calculation and output module.

TYPE DEFINITIONS

DayType	Record - Defines campaign day data.
Num	Integer - Campaign day.
CriteriaVal	Real - Phase activation criteria value.
N_AIMsn	Integer - Number of AI flight groups allocated for the campaign day.
N_FTRMsn	Integer - Number of fighter escort packages allocated for campaign day.
N_SEADMsn	Integer - Number of SEAD escort packages allocated for campaign day.
PhaseType	Record - Defines campaign phase data.
Active	Integer - 1 if phase is active, 0 otherwise.
TS	Array of reals - Phase total strength value for each strength category.
DS	Array of reals - Phase current strength for each strength category.
CS	Array of reals - Phase total strength value for each strength category.

DRS	Array of reals - Phase desired reduction in strength for each strength category.
ActiveStrength	Array of integer - 1 if strength category is active for phase, 0 otherwise.
ActiveEquip	Array of integer - 1 if equipment type is active for phase, 0 otherwise.
DL	Array of reals - Desired levels for each unit type and strength category.

AIMsntype	Record - Defines AI flight group mission data.
AB	Integer - Air base flight group originated from.
AC	Integer - Aircraft type of flight group.
Tgt	Integer - Target unit for flight group.
Equip	Integer - Equipment type flight group is to attack.
FTREsc	Integer - 1 if fighter escort package is attached, 0 otherwise.
SEADEsc	Integer - 1 if SEAD escort package is attached, 0 otherwise.
PSAA	Real - Probability of successfully flying to target due to enemy air-to-air defenses.
PSSA	Real - Probability of successfully flying to target due to enemy surface-to-air defenses.
Sorties	Integer - Number of aircraft in flight group.

Escorttype	Record - Defines fighter and SEAD escort package data.
AB	Integer - Air base escort package originated from.
AC	Integer - AI aircraft type escort package is supporting.
Tgt	Integer - Target unit for escort package.
Sorties	Integer - Number of escort packages assigned to target.

Tgttype	Record - Defines target unit data
ID	String - Name of target unit.
UnitType	Integer - Type of unit.
Node	Integer - Ground node target unit is located.
Grid	Integer - Air grid target node is located below.
GridX	Real - Air grid X location.
GridY	Real - Air grid Y location.
Echelon	Real - Echelon weight of target unit.
COA	Real - COA probability of target unit.
TS	Array of reals - Total strength of target unit for each strength category.
CS	Array of reals - Current strength of target unit for each strength category.

DS	Array of reals - Desired strength of target unit for each strength category.
DRS	Array of reals - Desired reduction of strength of target unit for each strength category.
TOE	Array of reals - TOE quantity of target unit for each equipment type.
CNE	Array of reals - Current quantity of target unit for each equipment type.
BlueABtype	Record - Defines friendly air base data.
ID	String - Name of friendly air base.
Grid	Integer - Air grid location of friendly air base.
SquadID	Array of integer - ID of squadron attached to friendly air base.
TOE	Array of integer - TOE quantity of aircraft attached to friendly air base.
CNE	Array of integer - Current quantity of aircraft attached to friendly air base.

CONSTANTS

NumGrids	Constant - Number of grids in air network.
NumRows	Constant - Number of rows in air grid.
NumCols	Constant - Number of columns in air grid.
GridWidth	Constant - Width of each grid in air grid.
NumPhases	Constant - Number of phases in campaign.
NumTgts	Constant - Number of target units.
NumUnittypes	Constant - Number of unit types.
NumEquip	Constant - Number of equipment.
NumBlueAB	Constant - Number of friendly air bases.
NumBlueAC	Constant - Number of friendly aircraft types.
NumMsn	Constant - Maximum number of flight group missions in a day.
NumStrengths	Constant - Number of strength categories.
DataDir	Constant - DOS directory of data files.
GAMSDir	Constant - DOS directory of GAMS files.
BlueAC	Array of integers - Maximum range and number of sorties per day for each friendly aircraft type.
PKASD	Array of reals - Probability of kill of an equipment type by each type of friendly aircraft type.
PKAAD	Array of reals - Maximum range and probability of kill of enemy fighter against each friendly aircraft type.

Equip	Array of reals - Strength category values for each equipment type and strength category.
EquipName	Array of strings - Names of each equipment type.
StrengthName	Array of strings - Three letter identifier for each strength category.
UnitTypeName	Array of strings - Names of each unit type.

VARIABLES

Day	Array of Daytype - Data for each campaign day.
Phase	Array of Phasetype - Data of each phase.
AIMsn	Array of AIMsntype - Data of each AI flight group mission.
FTRMsn	Array of Escorttype - Data of each fighter package mission.
SEADMsn	Array of Escorttype - Data of each SEAD package mission.
Tgt	Array of Tgttype - Data of each target unit.
BlueAB	Array of BlueABtype - Data of each friendly air base.
PKAA	Array of NumGrids - Stores the accumulated probability of kill due to enemy air-to-air threat for each air grid.
PKSA	Array of NumGrids - Stores the accumulated probability of kill due to enemy air-to-air threat for each air grid.
g	Linked list - Stores the optimal route calculated by the air route planning module.

FILE TYPE VARIABLES

Gridfile	Text - File of air grid coordinates.
Dayfile	Text - File of campaign day data.
Phasefile	Text - File of phase data.
AIMsnfile	Text - File of AI flight group mission data.
FTRMsnfile	Text - File of fighter package mission data.
SEADMsnfile	Text - File of SEAD package mission data.
Tgtfile	Text - File of target unit data.
BlueABfile	Text - File of friendly air base data.
DayResults	Text - File of campaign day output.
DRSfile	Text - File of desired reduction in strength data for each potential target formatted for air allocation MIP.
ECNfile	Text - File of current number of equipment for each potential target formatted for air allocation MIP.
PKfile	Text - File of probability of kills for each equipment and friendly aircraft type combination formatted for air allocation MIP.

ESVfile	Text - File of strength values of each equipment type and strength category combination formatted for air allocation MIP.
PSAAfile	Text - File of probability of success due to enemy air-to-air threat for each friendly aircraft type flying from a friendly air base to a target unit formatted for air allocation MIP.
PSSAfile	Text - File of probability of success due to enemy surface-to-air threat for each friendly aircraft type flying from a friendly air base to a target unit formatted for air allocation MIP.
RANGEfile	Text - File of maximum range constraints for each friendly aircraft type and target unit combination.

B. FILE OUTPUT

The following files are created by the Pascal program. Each run of the $(AA)^2$ model overwrites previous files of the same name.

IDENTIFIERS

< Day >	Current campaign day.
< Phase >	Campaign phase.
< DataDir >	DOS directory of data files.
< GAMSDir >	DOS directory of data files.

FILES

< DataDir > Dayfile.dat	Stores current day information.
< DataDir > Phasefile.dat	Stores desired levels for each phase.
< DataDir > Tgtfile < Day >.dat	Stores potential target information.
< DataDir > ABfile < Day >.dat	Stores number of aircraft located at each friendly air base.
< DataDir > COA < Day >.dat	Stores COA probabilities for use by sorties allocation MIP.
< DataDir > ECH < Day >.dat	Stores echelon values for use by sortie allocation MIP.
< DataDir > DRS < Day > < Phase >.dat	Stores desired reduction in strength values for use by sortie allocation MIP.
< DataDir > ECN < Day > < Phase >.dat	Stores current number of equipment systems of each target for sortie allocation MIP.

< DataDir > ESV < Phase >.dat	Stores equipment strength values for use by the sortie allocation MIP.
< DataDir > PK < Phase >.dat	Stores the probability of kill for each friendly aircraft type versus each equipment type.
< DataDir > PSAA < Day >.dat	Stores the probability of success due to enemy air-to-air threat for use by the sortie allocation MIP.
< DataDir > PSSA < Day >.dat	Stores the probability of success due to enemy surface-to-air threat for use by the sortie allocation MIP.
< DataDir > RANGE < Day >.dat	Stores the matrix of potential target ranges (1 or 0) for sortie allocation MIP.
< GAMSDir > AIMsn < Day >.dat	Stores the sortie allocation of AI missions read from the sortie allocation MIP output.
< GAMSDir > FTRMsn < Day >.dat	Stores the escort fighter package missions read from the sortie allocation MIP output.
< GAMSDir > SEADMsn < Day >.dat	Stores the escort SEAD package missions read from the sortie allocation MIP output.
<DataDir>PK<Phase>.dat	Stores the probability of kill of an equipment type by friendly aircraft types for use by te sortie allocation MIP.
<Data>AB<Day><Phase>.dat	Stores the number of AI sorties available from friendly air bases for use by the sortie allocation MIP.
<Data>MAXSEA<Day><Phase>.dat	Stores the number of SEAD packages available from friendly air bases for use by the sortie allocation MIP.
<Data>MAXFTR<Day><Phase>.dat	Stores the number of fighter packages available from friendly air bases for use by the sortie allocation MIP.

APPENDIX B. PASCAL SOURCE CODE

Appendix B contains the Pascal source code of the $(AA)^2$ model. The source code is available from Professor S. H. Parry, Naval Postgraduate School, Monterey California.

A. MAIN PROGRAM

```
{*************************************************************************
**
   Program:    Main.PAS
   Description: Main program for (AA)^2 model. Controls execution and calls
                procedures.
*************************************************************************
*}
program main;
uses Data, In_Out, BuildEADU, Strength, FlyMsn;
var     Day             : Daytype;
        Air_Grid        : grid_value;
        AIMsn           : AIMsndata;
        FTRMsn          : FTRMsndata;
        SEADMsn         : SEADMsndata;
        Phase           : Phasedata;
        Tgt             : Tgtdata;
        BlueAB          : BlueABdata;
begin
  randomize;
  InputData             (Day,Phase,Air_grid,AIMsn,FTRMsn,SEADMsn,Tgt,BlueAB);
  AttriteMission        (Day,AIMsn,FTRMsn,SEADMsn,Tgt,BlueAB);
  ComputeStrengths      (Day,Phase,Tgt);
  PhaseTransition       (Day,Phase,Tgt);
  OutputData            (Day,Phase,Tgt,BlueAB);
  BuildEAD              (Day,Air_Grid,Tgt,BlueAB);
end. {main}
```

B. DATA MODULE

```
{*************************************************************************
*}
        Program:        Data.PAS
        Description:    Defines variable types and constants used in the model.
```

```
{*************************************************************************
*}
unit Data;
interface
const   NumGrids      = 100;
        NumRows       = 10;
        NumCols       = 10;
        GridWidth     = 30;
        NumPhases     = 2;
        NumTgts       = 8;
        NumUnitTypes = 5;
        NumEquip      = 10;
        NumBlueAB     = 2;
        NumBlueAC     = 4;
        NumMsn        = 50;
        NumStrengths = 8;
        DataDir       = 'd:\thesis\data\';
        GamsDir       = 'd:\thesis\gams\';
type    Daytype = record
                Num             : integer;
                Criteria        : integer;
                CriteriaVal     : real;
                N_AIMsn         : integer;
                N_FTRMsn        : integer;
                N_SEADMsn       : integer;
              end; {Daytype}
        Phasetype = record
                Active  : integer;
                TS      : array [1..NumStrengths] of real;
                DS      : array [1..NumStrengths] of real;
                CS      : array [1..NumStrengths] of real;
                DRS     : array [1..NumStrengths] of real;
                ActiveStrength  :  array [1..NumStrengths] of integer;
                ActiveEquip             :  array [1..NumEquip]    of integer;
                DL              : array [0..NumUnitTypes,1..NumStrengths] of real;
              end; {PhaseType}
        AIMsntype = record
                AB              : integer;
                AC              : integer;
                Tgt             : integer;
                Equip                   : integer;
                FTREsc          : integer;
                SEADEsc         : integer;
```

```
        PSAA            : real;
        PSSA            : real;
        Sorties                 : integer;
      end; {Msntype}
Escorttype = record
        AB      : integer;
        AC      : integer;
        Tgt     : integer;
        Sorties         : integer;
      end; {Escorttype}
Tgttype = record
        ID              : string[12];
        UnitType        : integer;
        Node            : integer;
        Grid            : integer;
        GridX                   : real;
        GridY                   : real;
        Echelon         : real;
        COA                     : real;
        TS              : array [1..NumStrengths] of real;
        CS              : array [1..NumStrengths] of real;
        DS              : array [1..NumStrengths] of real;
        DRS             : array [1..NumStrengths] of real;
        TOE                     : array [1..NumEquip] of integer;
        CNE                     : array [1..NumEquip] of integer;
      end; {Tgttype}
BlueABType = record
        ID                      : string[12];
        Grid            : integer;
        SquadID         : array [1..NumBlueAC] of integer;
        TOE             : array [1..NumBlueAC] of integer;
        CNE             : array [1..NumBlueAC] of integer;
      end; {BlueABType}
Gridtype = record
      ax    : real;
      ay    : real;
      bx    : real;
      by    : real;
      cx    : real;
      cy    : real;
      dx    : real;
      dy    : real;
    end; {Gridtype}
```

```
        VertexPTR= ^AdjVertexType;
        AdjVertexType=record
                VertexNumber        : integer;
                Dis                 : real;
                Next                : VertexPTR;
              end; {AdjVertexType}
        VertexType = record
                visited     : boolean;
                Hardness    : real;
                PKSA        : real;
                PKAA        : real;
                RouteDis    : real;
                next_choice : integer;
                AdjVertexList: VertexPTR;
             end; {VertexType}
        VertexList          = array [1..NumGrids] of VertexType;
        PKAAdata            = array [1..NumBlueAC + 1] of real;
        PKASdata            = array [1..NumBlueAC, 1..NumEquip] of real;
        PKSAdata            = array [1..2, 1..NumBlueAC + 1] of real;
        Keep_value          = array [1..NumGrids] of real;
        Grid_value          = array [1..NumGrids] of gridtype ;
        AIMsndata           = array [1..NumMsn] of AIMsntype;
        FTRMsndata          = array [1..NumMsn] of Escorttype;
        SEADMsndata         = array [1..NumMsn] of Escorttype;
        Phasedata           = array [1..NumPhases] of Phasetype;
        Tgtdata             = array [1..NumTgts] of Tgttype;
        Equipdata1          = array [1..NumEquip, 1..NumStrengths] of real;
        Equipdata2          = array [1..NumEquip] of string;
        BlueABdata          = array [1..NumBlueAB] of BlueABType;
        BlueACdata          = array [1..NumBlueAC, 1..2] of integer;
        Strengthdata        = array [1..NumStrengths] of string;
        UnitTypedata        = array [1..NumUnitTypes] of string;
const
  BlueAC: BlueACdata =
{A6E   - A1} ((350 , 3),
{F-15E  - A2}  (280 , 4),
{F-18C  - A3}  (250 , 4),
{F-117A - A4}  (250 , 2));
  PKASD: PKASdata =
          {   E1 ,  E2 ,  E3 ,  E4 ,  E5 ,  E6 ,  E7 ,  E8 ,  E9 , E10 }
{A6  - A1}((0.225,0.441,0.441,0.441,0.378,0.225,0.315,0.315,0.567,0.441),
{F15 - A2} (0.270,0.473,0.473,0.473,0.405,0.225,0.338,0.338,0.608,0.473),
{F18 - A3} (0.216,0.432,0.432,0.432,0.432,0.243,0.324,0.360,0.648,0.468),
```

```
{F117- A4} (0.135,0.168,0.473,0.473,0.405,0.113,0.338,0.338,0.068,0.473));
                                  { Rng,    A1 ,   A2 ,   A3 ,   A4  }
  PKAAD:PKAAdata ={Fighter} ( 100.0, 0.0630,  0.03150, 0.03150, 0.00038);
                                   { Rng,    A1 ,   A2 ,   A3 ,   A4  }
  PKSAD:PKSAdata ={LR SAM} (( 60.0,  0.1125,  0.0900,  0.0900,  0.0009 ),
                          {SR SAM}  ( 10.0,  0.1140,  0.0900,  0.0900,  0.00090 ));
                                {STRENGTH CATEGORIES}
  Equip:Equipdata1 =
             {EQUIP}    { AAA,  AAS,  ASA,  ASS,  CC2,  MOB,  LPO,  LST }
          {TANK - E1} (( 0.0,  0.0,  0.0, 10.0,  0.0,  0.0,  0.0,  0.0 ),
          {ARTY - E2}  ( 0.0,  0.0,  0.0, 12.0,  0.0,  0.0,  0.0,  0.0 ),
        {ATTJET - E3} ( 2.0, 12.0,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0 ),
        {FITJET - E4}  (11.0,  4.5,  0.0,  0.0,  0.0,  0.0,  0.0,  0.0 ),
        {LR SAM - E5} ( 0.0,  0.0,  7.0,  0.0,  0.0,  0.0,  0.0,  0.0 ),
        {SR SAM - E6} ( 0.0,  0.0,  5.0,  0.0,  0.0,  0.0,  0.0,  0.0 ),
        {C2 Van - E7}  ( 0.0,  0.0,  0.0,  0.0,  2.5,  0.0,  0.0,  0.0 ),
       {RWY SEC - E8}( 0.0,  0.0,  0.0,  0.0,  0.0,  2.0,  0.0,  0.0 ),
      {POL BLIV - E9} ( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  2.0,  0.0 ),
   { POL TRUCK - E10}( 0.0,  0.0,  0.0,  0.0,  0.0,  0.0,  2.0,  5.0 ));
  EquipName:Equipdata2 = ('TANK', 'ARTY', 'ATK JET', 'FTR JET', 'LR SAM',
                          'SR SAM', 'C2 VAN', 'RWY SEC', 'POL BLIVET',
                          'POL TRUCK');
  StrengthName:Strengthdata = ('AAA', 'AAS', 'ASA', 'ASS', 'CC2', 'MOB',
                               'LPO','LST');
  UnitTypeName:UnitTypedata = ('AIR BASE    ', 'MECH BDE    ', 'ARTILLARY
                               BTN',  'LOGISTIC UNIT', 'LOGISTIC BASE');
implementation
end. {Module Data}
```

C. INPUT AND OUTPUT MODULE

```
{*************************************************************************
**
   Module:      In_Out.PAS
   Description:  Creates Day Results file, retreives data stored in text files
            from previous day, writes data at end of day to files.
*************************************************************************
*}
Unit In_Out ;
```

```
interface
uses Dos,Data;
Procedure InputData  (var Day        : Daytype;
                      var Phase      : Phasedata;
                      var Air_Grid   : Grid_value;
                      var AIMsn      : AIMsndata;
                      var FTRMsn     : FTRMsndata;
                      var SEADMsn    : SEADMsndata;
                      var Tgt        : Tgtdata;
                      var BlueAB     : BlueABdata);
Procedure OutputData       (var Day        : Daytype;
                            var Phase      : Phasedata;
                            var Tgt        : Tgtdata;
                            var BlueAB     : BlueABdata);
implementation
{** Procedure Input data
*************************************************}
Procedure InputData ;
  var   i,j,k,s                       : integer;
        Unittype,Code                 : integer;
        DayCh,PhaseCh,FileStr         : String;
        Year,Month,Dy,DayofWeek,
        Hour,Minute,Second,Sec100     : Word;
        Gridfile,Dayfile,
        AIMsnfile,FTRMsnfile,
        SEADMsnfile,Tgtfile,
      BlueABfile,Phasefile,
      PKEADfile,PSAAfile,PSSAfile,
      DRSfile,DayResults              : text;
      PSAA     : array [1..NumBlueAB,1..NumBlueAC,1..NumTgts,0..1] of real;
      PSSA     : array [1..NumBlueAB,1..NumBlueAC,1..NumTgts,0..1] of real;
begin
{** Day input
*********************************************************}
  assign (Dayfile,DataDir+'Dayfile.dat');
  reset  (Dayfile);
  readln (Dayfile,Day.Num,Day.CriteriaVal);
  while not (eof(Dayfile)) do begin
    readln (Dayfile, i, Phase[i].Active);
  end; {while}
  close  (Dayfile);
  Str(Day.Num,DayCh);
  assign  (DayResults,DataDir+'DayRes.dat');
```

```
  if (Day.Num = 0) then rewrite (DayResults)
  else append (DayResults);
  writeln (DayResults,'*****************************************************');
  writeln (DayResults,'*       AIR ALLOCATION / AIR APPORTIONMENT          *');
  writeln (DayResults,'*              (AA)^2                               *');
  writeln (DayResults,'*               MODEL                               *');
  writeln (DayResults,'*****************************************************');
  GetDate (Year,Month,Dy,Dayofweek);
  GetTime (Hour,Minute,Second,Sec100);
  writeln (DayResults,'*  Campaign Day: ',Day.Num);
  writeln (DayResults,'*  Todays Date : ',Month,'/',Dy,'/',Year);
  writeln (DayResults,'*  Todays Time : ',Hour:2,':',Minute:2,':',Second:2);
  writeln (DayResults,'*  Random Seed : ',RandSeed);
  writeln (DayResults,'*****************************************************');
  writeln ('*****************************************************');
  writeln ('*       AIR ALLOCATION / AIR APPORTIONMENT          *');
  writeln ('*              (AA)^2                               *');
  writeln ('*               MODEL                               *');
  writeln ('*****************************************************');
  writeln ('*  Campaign Day: ',Day.Num);
  writeln ('*  Todays Date : ',Month,'/',Dy,'/',Year);
  writeln ('*  Todays Time : ',Hour:2,':',Minute:2,':',Second:2);
  writeln ('*  Random Seed : ',RandSeed);
  writeln ('*****************************************************');
{** Grid coordinate input
************************************************}
  assign (Gridfile,DataDir+'g10x1030.dat');
  reset  (Gridfile);
  while not(eof(Gridfile)) do begin
    readln(Gridfile,i, air_grid[i].ax, air_grid[i].ay,
                   air_grid[i].bx, air_grid[i].by,
                   air_grid[i].cx, air_grid[i].cy,
                   air_grid[i].dx, air_grid[i].dy );
  end; {while}
  writeln ('**  Grid coordinates inputed.');
  close  (Gridfile);
{** Phase input
************************************************************}
  assign (Phasefile,DataDir+'Phasefile.dat');
  reset  (Phasefile);
  while not(eof(Phasefile)) do begin
    readln  (Phasefile,i);
    writeln (DayResults,'*************************************************');
```

```
    writeln (DayResults,'  Phase: ',i,' Desired Levels');
    write   (DayResults,'    Phase ',i,' is ');
    if (Phase[i].Active = 1) then writeln (DayResults,'active.')
    else writeln (DayResults,' not active.');
    write   (DayResults,'Unit Type     ');
    for s := 1 to NumStrengths do begin
      write (DayResults,StrengthName[s]:5);
    end; {for s}
    writeln (DayResults);
    for j := 1 to NumUnittypes do begin
      read  (Phasefile,Unittype);
      write (DayResults,UnittypeName[Unittype]:9);
      for k := 1 to NumStrengths do begin
        read  (Phasefile,Phase[i].DL[UnitType,k]);
        write (DayResults,Phase[i].DL[Unittype,k]:5:1);
      end; {for k}
      readln  (Phasefile);
      writeln (DayResults);
    end; {for j}
    writeln ('**  Phase: ',i,' desired levels inputed.');
  end; {while}
  close (Phasefile);
{** Target input ************************************************************}
  assign  (Tgtfile,DataDir+'Tgtfile'+DayCh+'.dat');
  reset   (Tgtfile);
  writeln (DayResults);
  writeln (DayResults,'************************************************');
  writeln (DayResults,'  Target Input');
  writeln (DayResults);
  write   (DayResults,'Tgt#  TgtID      UnitType     Node Grid   X    Y   Ech ');
  writeln (DayResults,'  COA');
  while not(eof(Tgtfile)) do begin
    readln  (Tgtfile,i,Tgt[i].ID,Tgt[i].UnitType,Tgt[i].Node,Tgt[i].Grid,
                  Tgt[i].GridX,Tgt[i].GridY,Tgt[i].Echelon,Tgt[i].COA);
    writeln (DayResults,i:2,Tgt[i].ID:12,UnittypeName[Tgt[i].UnitType]:15,
                   Tgt[i].Node:6,Tgt[i].Grid:4,Tgt[i].GridX:6:1,
                   Tgt[i].GridY:6:1,Tgt[i].Echelon:4:1,Tgt[i].COA:6:1);
    for j := 1 to NumEquip do begin
      readln  (Tgtfile,k,Tgt[i].TOE[j],Tgt[i].CNE[j]);
    end; {for j}
  end; {while}
  close (Tgtfile);
  writeln ('**  Target data inputed.');
```

```
{** BlueAirBase Input ************************************************}
  assign (BlueABfile,DataDir+'ABfile'+DayCh+'.dat');
  reset  (BlueABfile);
  writeln (DayResults);
  writeln (DayResults,'**********************************');
  writeln (DayResults,'  Blue Air Base Input');
  writeln (DayResults,' #    ID       Grid');
  while not(eof(BlueABfile)) do begin
    readln  (BlueABfile,i,BlueAB[i].ID,BlueAB[i].Grid);
    writeln (DayResults,i:2,BlueAB[i].ID:10,BlueAB[i].Grid:4);
    writeln (DayResults,'  Squad  TOE  Curr');
    for j := 1 to NumBlueAC do begin
      readln  (BlueABfile,BlueAB[i].SquadID[j],BlueAB[i].TOE[j],
            BlueAB[i].CNE[j]);
      writeln (DayResults,BlueAB[i].SquadID[j]:6,BlueAB[i].TOE[j]:6,
                  BlueAB[i].CNE[j]:6);
    end; {for j}
  end; {while}
  close (BlueABfile);
  writeln ('**  Blue Air Base data inputed.');
{** Enemy Air Defense Input ******************************************}
  assign (PSAAfile,DataDir+'PSAA'+DayCh+'.dat');
  reset  (PSAAfile);
  readln (PSAAfile);
  while not(eof(PSAAfile)) do begin
    readln (PSAAfile,FileStr);
    Val(Copy(FileStr,2,1),i,Code);
    Val(Copy(FileStr,5,1),j,Code);
    Val(Copy(FileStr,8,1),k,Code);
    Val(Copy(FileStr,14,7),PSAA[i,j,k,0],Code);
    Val(Copy(FileStr,26,7),PSAA[i,j,k,1],Code);
  end; {while}
  close (PSAAfile);
  assign (PSSAfile,DataDir+'PSSA'+DayCh+'.dat');
  reset  (PSSAfile);
  readln (PSSAfile);
  while not(eof(PSSAfile)) do begin
    readln (PSSAfile,FileStr);
    Val(Copy(FileStr,2,1),i,Code);
    Val(Copy(FileStr,5,1),j,Code);
    Val(Copy(FileStr,8,1),k,Code);
    Val(Copy(FileStr,14,7),PSSA[i,j,k,0],Code);
    Val(Copy(FileStr,26,7),PSSA[i,j,k,1],Code);
```

```
  end; {while}
  close (PSSAfile);
  writeln ('** Enemy Air Defense data inputed.');
{** AI Mission input  ************************************************}
  assign (AIMsnfile,GamsDir+'Msnfile'+DayCh+'.dat');
  reset  (AIMsnfile);
  writeln (DayResults);
  i := 0;
  while not(eof(AIMsnfile)) do begin
    inc(i);
    readln (AIMsnfile,FileStr);
    Val(Copy(FileStr,2,1),AIMsn[i].AB,Code);
    Val(Copy(FileStr,5,1),AIMsn[i].AC,Code);
    Val(Copy(FileStr,8,1),AIMsn[i].Tgt,Code);
    Val(Copy(FileStr,11,2),AIMsn[i].Equip,Code);
    Val(Copy(FileStr,16,1),AIMsn[i].FTREsc,Code);
    Val(Copy(FileStr,20,1),AIMsn[i].SEADEsc,Code);
    Val(Copy(FileStr,21,30),AIMsn[i].Sorties,Code);
    AIMsn[i].PSAA :=
PSAA[AIMsn[i].AB,AIMsn[i].AC,AIMsn[i].Tgt,AIMsn[i].FTREsc];
    AIMsn[i].PSSA :=
PSSA[AIMsn[i].AB,AIMsn[i].AC,AIMsn[i].Tgt,AIMsn[i].SEADEsc];
  end; {while}
  Day.N_AIMsn := i;
  close  (AIMsnfile);
  writeln ('** Air Mission data inputed.');
{** Fighter Escort input
************************************************}
  assign (FtrMsnfile,GamsDir+'FtrMsn'+DayCh+'.dat');
  reset  (FtrMsnfile);
  i := 0;
  while not(eof(FtrMsnfile)) do begin
    inc(i);
    readln (FtrMsnfile,FileStr);
    Val(Copy(FileStr,2,1),FtrMsn[i].AB,Code);
    Val(Copy(FileStr,5,1),FtrMsn[i].AC,Code);
    Val(Copy(FileStr,8,1),FtrMsn[i].Tgt,Code);
    Val(Copy(FileStr,9,30),FtrMsn[i].Sorties,Code);
  end; {while}
  Day.N_FtrMsn := i;
  close  (FtrMsnfile);
  writeln ('** Fighter Mission data inputed.');
```

```
{** SEAD Mission input
*************************************************}
  assign (SEADMsnfile,GamsDir+'SEADMsn'+DayCh+'.dat');
  reset  (SEADMsnfile);
  i := 0;
  while not(eof(SEADMsnfile)) do begin
    inc(i);
    readln (SEADMsnfile,FileStr);
    Val(Copy(FileStr,2,1),SEADMsn[i].AB,Code);
    Val(Copy(FileStr,5,1),SEADMsn[i].AC,Code);
    Val(Copy(FileStr,8,1),SEADMsn[i].Tgt,Code);
    Val(Copy(FileStr,9,30),SEADMsn[i].Sorties,Code);
  end; {while}
  Day.N_SEADMsn := i;
  close  (SEADMsnfile);
  close  (DayResults);
  writeln ('**  SEAD Mission data inputed.');
end; {InputData}
{**  OutputData ***********************************************************}
procedure OutputData;
var i,j,k,s         : integer;
   DayCh         : string;
   Dayfile,
   Phasefile,
   Tgtfile,
   Updatefile,
   COAfile,
   ECHfile,
   BlueABfile,
   BlueABtable    : text;
begin
{** Day output
*********************************************************}
  assign  (Dayfile,DataDir+'Dayfile.dat');
  rewrite (Dayfile);
  writeln (Dayfile,Day.Num+1:3,Day.CriteriaVal:8:2);
  for i := 1 to NumPhases do begin
    writeln (Dayfile,i:6,Phase[i].Active:6);
  end; {for}
  close   (Dayfile);
  str(Day.Num+1,DayCh);
  writeln ('**  Campaign Day data Outputed.');
```

```
{** Target
output*****************************************************}
  assign  (Updatefile,DataDir+'Update'+DayCh+'.dat');
  reset   (Updatefile);
  assign  (Tgtfile,DataDir+'Tgtfile'+DayCh+'.dat');
  rewrite (Tgtfile);
  for i := 1 to NumTgts do begin
    readln  (Updatefile, i,Tgt[i].ID,Tgt[i].UnitType,Tgt[i].Node,Tgt[i].Grid,
                Tgt[i].GridX,Tgt[i].GridY,Tgt[i].Echelon,Tgt[i].COA);
    writeln (Tgtfile,i, Tgt[i].ID:4,Tgt[i].UnitType:4,Tgt[i].Node:4,Tgt[i].Grid:4,
                Tgt[i].GridX:6:1,Tgt[i].GridY:6:1,Tgt[i].Echelon:2,Tgt[i].COA:6:1);
    for j := 1 to NumEquip do begin
      writeln (Tgtfile,j:4,tgt[i].toe[j]:4,tgt[i].CNE[j]:4);
    end; {for j}
  end; {for i}
  close (Tgtfile);
  close (Updatefile);
  writeln ('**  Target data updated and outputed.');
{**  Course of Action (COA) and Echelon (ECH) output  ************************}
  assign  (COAfile,DataDir+'COA'+DayCh+'.dat');
  assign  (ECHfile,DataDir+'ECH'+DayCh+'.dat');
  rewrite (COAfile);
  rewrite (ECHfile);
  write (COAfile,'/');
  write (ECHfile,'/');
  for i := 1 to NumTgts do begin
    writeln (COAfile,' T',i,'    ',Tgt[i].COA:5:1);
    writeln (ECHfile,' T',i,'    ',Tgt[i].Echelon:5:1);
  end; {for i}
  write (COAfile,'/');
  write (ECHfile,'/');
  close (COAfile);
  close (ECHfile);
{**  BlueAB output
*******************************************************}
  assign  (BlueABfile,DataDir+'ABfile'+DayCh+'.dat');
  assign  (BlueABtable,DataDir+'AB'+DayCh+'1.dat');
  rewrite (BlueABfile);
  rewrite (BlueABtable);
  writeln (BlueABtable,'    A1   A2   A3   A4');
  for i := 1 to NumBlueAB do begin
    writeln (BlueABfile,i,BlueAB[i].ID:12,BlueAB[i].Grid:4);
    write   (BlueABtable,'B',i);
```

```
    for j := 1 to NumBlueAC do begin
      writeln (BlueABfile,BlueAB[i].SquadID[j]:6,BlueAB[i].TOE[j]:4,
            BlueAB[i].CNE[j]:4);
      write  (BlueABtable,BlueAB[i].CNE[j]*BlueAC[j,2]:5);
    end; {for j}
    writeln (BlueABtable);
  end; {for i}
  close (BlueABfile);
  close (BlueABtable);
  writeln ('**  Blue Air Base data outputed.');
end; {OutputData}
end. {In_Out}
```

D. ATTRITION MODULE

```
{*************************************************************************
   Module:      FlyMsn.PAS
   Description:  Attrites enemy air defenses by SEAD escort package missions,
            attrites friendly AI sorties by enemy air defenses, attrites
            enemy targets by friendly AI sorties, and outputs results to
            Day Result file.
*************************************************************************
*}
unit FlyMsn ;
interface
uses Data,PKTool1;
type
  storetype = array [1..20] of integer;
procedure AttriteMission    (var Day       : Daytype;
                             var AIMsn     : AIMsndata;
                             var FTRMsn    : FTRMsndata;
                             var SEADMsn   : SEADMsndata;
                             var Tgt       : Tgtdata;
                             var BlueAB    : BlueABdata);
function  Fact(x: integer):real;
function  Bin (n: integer; p: real):integer;
implementation
{**  Procedure AttriteMission
*********************************************}
Procedure AttriteMission;
var     i,j,k,ACKilled,EquipKilled,
        SAMKills                    : integer;
```

```
        Pkill                   : real;
        MaxSorties              : array [1..NumBlueAB,1..NumBlueAC] of integer;
        Sorties                 : array [1..NumBlueAB,1..NumBlueAC] of integer;
        ACKills                 : array [1..NumBlueAB,1..NumBlueAC] of integer;
        EquipKills              : array [1..Numequip]  of integer;
        Fight                   : array [1..NumBlueAB] of integer;
        SEAD                    : array [1..NumBlueAB] of integer;
        SumMaxSorties           : integer;
        SumSorties              : integer;
        SumACKills              : integer;
        DayCh                   : string;
        DayResults              : text;
begin
  SumMaxSorties := 0;
  SumSorties    := 0;
  SumACKIls     := 0;
  Str(Day.Num,DayCh);
  assign  (DayResults,DataDir+'DayRes.dat');
  append  (DayResults);
{**  SEAD Escort Mission Calculations
*************************************}
  writeln (DayResults,'*************************************************');
  writeln (DayResults,'   SEAD Mission Results');
  for i := 1 to Day.N_SEADMsn do begin
    writeln (DayResults,'MSN#  Air Base   A/C     TGT   SORTIES ');
    writeln
(DayResults,i:3,BlueAB[SEADMsn[i].AB].SquadID[SEADMsn[i].AC]:5,Tgt[SEADMsn
[i].Tgt].ID:15,
                SEADMsn[i].Sorties:4);
    if (Tgt[SEADMsn[i].Tgt].CNE[5] > 0) then begin
      SAMKills := bin(SEADMsn[i].Sorties*2,0.15);
      if (SAMKills > Tgt[SEADMsn[i].Tgt].CNE[5]) then begin
        SAMKills := Tgt[SEADMsn[i].Tgt].CNE[5];
        Tgt[SEADMsn[i].Tgt].CNE[5] := 0;
        writeln (DayResults,'  ',EquipName[5],' killed by SEAD: ',SAMKills);
      end {if}
      else begin
        Tgt[SEADMsn[i].Tgt].CNE[5] := Tgt[SEADMsn[i].Tgt].CNE[5] -
                                SAMKills;
        writeln (DayResults,'  ',EquipName[5],' killed by SEAD: ',SAMKills);
      end {else}
    end {if}
    else begin
```

```
      if (Tgt[SEADMsn[i].Tgt].CNE[6] > 0) then begin
        SAMKills := bin(SEADMsn[i].Sorties*2,0.10);
        if (SAMKills > Tgt[SEADMsn[i].Tgt].CNE[6]) then begin
          SAMKills := Tgt[SEADMsn[i].Tgt].CNE[6];
          Tgt[SEADMsn[i].Tgt].CNE[6] := 0;
          writeln (Dayresults,' ',EquipName[6],' killed by SEAD: ',SAMKills);
        end {if}
        else begin
          Tgt[SEADMsn[i].Tgt].CNE[6] := Tgt[SEADMsn[i].Tgt].CNE[6] -
                                        SAMKills;
          writeln (DayResults,' ',EquipName[6],' killed by SEAD: ',SAMKills);
        end {else}
      end; {if}
    end; {else}
  end; {for i}
{** Fighter Escort Output
************************************************}
  writeln
(DayResults,'*****************************************************');
  writeln (DayResults,'    Fighter Escort Missions');
  writeln (DayResults,' MSN# Air Base  A/C Esc  TGT  Sorties');
  for i := 1 to Day.N_FTRMsn do begin
    writeln
(DayResults,i:3,BlueAB[FTRMsn[i].AB].ID,BlueAB[FTRMsn[i].AB].SquadID[FTRMsn[
i].AC],
                  Tgt[FTRMsn[i].Tgt].ID,FTRMsn[i].Sorties:4);
  end; {for i}
{** AI Mission Calculations  ****************************************}
  writeln (DayResults,'*****************************************************');
  writeln (DayResults,'  AI Mission Results');
  for i := 1 to NumBlueAB do begin
    for j := 1 to NumBlueAC do begin
      MaxSorties[i,j] := BlueAB[i].CNE[j] * BlueAC[j,2];
      SumMaxSorties   := SumMaxSorties + MaxSorties[i,j];
      Sorties[i,j]    := 0;
      ACKills[i,j]    := 0;
    end;{for j}
  end; {for i}
  for i := 1 to NumEquip do begin
    EquipKills[i] := 0;
  end; {for i}
  for i := 1 to Day.N_AIMsn do begin
```

```
    writeln (DayResults,'MSN# A/C       TGT  EF  ES PSAA PSSA  SORTIES
EQUIP');
    write
(DayResults,i:3,BlueAB[AIMsn[i].AB].SquadID[AIMsn[i].AC]:5,Tgt[AIMsn[i].Tgt].ID:
15);
    if (AIMsn[i].FTREsc = 1) then write (DayResults,'YES':4)
    else write (DayResults,'NO':4);
    if (AIMsn[i].SEADEsc = 1) then write (DayResults,'Yes':4)
    else write (DayResults,'NO':4);
    writeln (DayResults,AIMsn[i].PSAA:7:4,AIMsn[i].PSSA:7:4,AIMsn[i].Sorties:4,
          EquipName[AIMsn[i].Equip]:12);
    Sorties[AIMsn[i].AB,AIMsn[i].AC] := Sorties[AIMsn[i].AB,AIMsn[i].AC] +
AIMsn[i].Sorties;
    PKill := 1 - (AIMsn[i].PSAA*AIMsn[i].PSSA);
    ACKilled := Bin(AIMsn[i].Sorties,PKill);
    AIMsn[i].Sorties := AIMsn[i].Sorties - ACKilled;
    BlueAB[AIMsn[i].AB].CNE[AIMsn[i].AC] :=
          BlueAB[AIMsn[i].AB].CNE[AIMsn[i].AC] - ACKilled;
    if (BlueAB[AIMsn[i].AB].CNE[AIMsn[i].AC] < 0) then
      BlueAB[AIMsn[i].AB].CNE[AIMsn[i].AC] := 0;
    EquipKilled := bin(AIMsn[i].Sorties,PKASD[AIMsn[i].AC,AIMsn[i].Equip]);
    if (EquipKilled > Tgt[AIMsn[i].Tgt].CNE[AIMsn[i].Equip]) then
       EquipKilled := Tgt[AIMsn[i].Tgt].CNE[AIMsn[i].Equip];
    Tgt[AIMsn[i].Tgt].CNE[AIMsn[i].Equip]:=
            Tgt[AIMsn[i].Tgt].CNE[AIMsn[i].Equip] - EquipKilled;
    writeln (DayResults,'  PK of Enemy AD: ',PKill:8:4);
    writeln (DayResults,'  PK for ',EquipName[AIMsn[i].Equip],':  ',
                  PKASD[AIMsn[i].AC,AIMsn[i].Equip]:6:4);
    writeln (DayResults,'  **  A/C killed by EAD: ',ACKilled);
    writeln (DayResults,'  **  ',EquipName[AIMsn[i].Equip],
                ' destroyed: ',EquipKilled);
    writeln (DayResults,'*******************************************');
    ACKills[AIMsn[i].AB,AIMsn[i].AC] := ACKills[AIMsn[i].AB,AIMsn[i].AC] +
ACKilled;
    EquipKills[AIMsn[i].Equip]    := EquipKills[AIMsn[i].Equip] + EquipKilled;
  end; {for i}
{**  Output to Day Results
*********************************************}
  writeln (DayResults,'************************************************');
  writeln (DayResults,'        Summary Statistics');
  writeln (DayResults,'************************************************');
  writeln (DayResults,'     Sorties Available');
  writeln (DayResults,'             A-6   F-15   F-18   F-117');
```

```
for i := 1 to NumBlueAB do begin
  write (DayResults,BlueAB[i].ID);
  for j := 1 to NumBlueAC do begin
    if (MaxSorties[i,j] > 0) then
      write (DayResults,(MaxSorties[i,j]):7)
    else write (DayResults,'N/A':7);
  end; {for j}
  writeln (Dayresults);
end; {for i}
writeln (DayResults,'  Total Sorties Available: ',SumMaxSorties:5);
writeln (DayResults,'     Sorties Used');
writeln (DayResults,'              A-6   F-15   F-18   F-117');
for i := 1 to NumBlueAB do begin
  write (DayResults,BlueAB[i].ID);
  for j := 1 to NumBlueAC do begin
    if (Sorties[i,j] > 0) then
      write (DayResults,(Sorties[i,j]):7)
    else write (DayResults,'N/A':7);
    SumSorties := SumSorties + Sorties[i,j];
  end; {for j}
  writeln (Dayresults);
end; {for i}
writeln (DayResults,'  Total Sorties Used: ',SumSorties:5);
writeln (DayResults,'     % Utilization');
writeln (DayResults,'              A-6   F-15   F-18   F-117');
for i := 1 to NumBlueAB do begin
  write (DayResults,BlueAB[i].ID);
  for j := 1 to NumBlueAC do begin
    if (MaxSorties[i,j] > 0) then
      write (DayResults,(Sorties[i,j]/MaxSorties[i,j]*100):7:0)
    else write (DayResults,'N/A':7);
  end; {for j}
  writeln (DayResults);
end; {for i}
writeln (DayResults,'  % Average Utilization:  ',(SumSorties/SumMaxSorties)*100:6:1);
writeln (DayResults,'     Attrition');
writeln (DayResults,'              A-6   F-15   F-18   F-117');
for i := 1 to NumBlueAB do begin
  write (DayResults,BlueAB[i].ID);
  for j := 1 to NumBlueAC do begin
    if (Sorties[i,j] > 0) then
      write (DayResults,(ACKills[i,j]):7)
    else write (DayResults,'N/A':7);
```

```
      SumACKIlls := SumACKills + ACKills[i,j];
    end; {for j}
    writeln (DayResults);
  end; {for i}
  writeln (DayResults,'  Total Attrition: ',SumACKills:3);
  writeln (DayResults,'     % Attrition');
  writeln (DayResults,'              A-6   F-15   F-18   F-117');
  for i := 1 to NumBlueAB do begin
    write (DayResults,BlueAB[i].ID);
    for j := 1 to NumBlueAC do begin
      if (Sorties[i,j] > 0) then
        write (DayResults,(ACKills[i,j]/Sorties[i,j]*100):7:0)
      else write (DayResults,'N/A':7);
    end; {for j}
    writeln (Dayresults);
  end; {for i}
  if (SumSorties > 0) then
    writeln (DayResults,'  % Average Attrition: ',(SumACKIlls/SumSorties)*100:6:1);
  writeln (DayResults,'    TOE Equipment');
  write   (DayResults,'          ');
  for i := 1 to NumEquip do begin
    write (DayResults,i:4);
  end; {for i}
  writeln (DayResults);
  for i := 1 to NumTgts do begin
    write (DayResults,Tgt[i].ID);
    for j := 1 to NumEquip do begin
      if (Tgt[i].TOE[j] > 0) then
        write (DayResults,Tgt[i].TOE[j]:4)
      else
        write (DayResults,'    ');
    end; {for j}
    writeln (DayResults);
  end; {for i}
  writeln (DayResults,'    Current Equipment');
  write   (DayResults,'          ');
  for i := 1 to NumEquip do begin
    write (DayResults,i:4);
  end; {for i}
  writeln (DayResults);
  for i := 1 to NumTgts do begin
    write (DayResults,Tgt[i].ID);
    for j := 1 to NumEquip do begin
```

```
        if (Tgt[i].TOE[j] > 0) then
          write (DayResults,Tgt[i].CNE[j]:4)
        else
          write (DayResults,'    ');
      end; {for j}
      writeln (DayResults);
    end; {for i}
    writeln (DayResults,'    % remaining Equipment');
    write   (DayResults,'          ');
    for i := 1 to NumEquip do begin
      write (DayResults,i:7);
    end; {for i}
    writeln (DayResults);
    for i := 1 to NumTgts do begin
      write (DayResults,Tgt[i].ID);
      for j := 1 to NumEquip do begin
        if (Tgt[i].TOE[j] > 0) then
          write (DayResults,((Tgt[i].CNE[j]/Tgt[i].TOE[j])*100):7:1)
        else
          write (DayResults,'       ');
      end; {for j}
      writeln (DayResults);
    end; {for i}
    close   (DayResults);
    writeln ('**  Air Missions flown and data collected.');
  end; {procedure AttriteMission}
  {**  Fact Function
  ********************************************************}
  function Fact;
  var  Product: real;
  begin
    Product := 1;
    if x > 0 then
      for i := 1 to x do begin
        Product := Product * i;
      end; {for}
    Fact := Product;
  end; {Fact}
  {**  Bin Function
  ********************************************************}
  function Bin;
  var  R,BinP,CumP  : real;
       x            : integer;
```

```
begin
  x := -1;
  R := Random;
  BinP := 0.0;
  CumP := 0.0;
  if P > 0 then begin
    repeat
      inc(x);
      BinP := (Fact(n)/(Fact(x)*Fact(n-x)))*exp(x*ln(p))*exp((n-x)*ln(1-p));
      CumP := CumP + BinP;
    until (R <= CumP) or (x = n);
  end {if}
  else x := 0;
  Bin := x;
end; {Bin}
end. {FlyMsn}
```

E. TARGET STRENGTH AND PHASE ACTIVATION MODULE

```
{*************************************************************************
**
   Module:      Strength.PAS
   Description: Computes the strength of all targets for strength categories
                of all phases.  Creates files used by air allocation program,
                and determines whether to activate next phase.
*************************************************************************
*}
unit Strength;
interface
uses Data;
procedure ComputeStrengths(var Day:      Daytype;
                           var Phase:    Phasedata;
                           var Tgt:      Tgtdata);
procedure PhaseTransition  (var Day:     Daytype;
                           var Phase:    Phasedata;
                           var Tgt:      Tgtdata);
implementation
{** Procedure ComputeStrengths
*****************************************}
procedure ComputeStrengths;
var    i,j,k,s,n              : integer;
       ETS,ECS                : real;
```

```
        DRSfile,
        ECNfile,
        PKfile,
        ESVfile              : text;
        DayCh,PhaseCh        : string;
begin
  {Initialize ActiveStrength and ActiveEquip matrices to not active}
  for i := 1 to NumPhases do begin
    for s := 1 to NumStrengths do begin
      Phase[i].ActiveStrength[s] := 0;
      for k := 1 to NumEquip do begin
        Phase[i].ActiveEquip[k] := 0;
      end; {for k}
    end; {for s}
  end; {for i}
  {Compute strengths for all active target, strength, and equipment combonations}
  for i := 1 to NumPhases do begin
      for s := 1 to NumStrengths do begin
        for j := 1 to NumTgts do begin
          if (Phase[i].DL[Tgt[j].UnitType,s] < 1.0) then begin
            Phase[i].ActiveStrength[s] := 1;
            ETS := 0.0;
            ECS := 0.0;
            for k := 1 to NumEquip do begin
              ETS := ETS + Tgt[j].TOE[k]*Equip[k,s];
              ECS := ECS + Tgt[j].CNE[k]*Equip[k,s];
              if (Phase[i].ActiveStrength[s]=1) and (Tgt[j].TOE[k]*Equip[k,s] > 0.0)
                then Phase[i].ActiveEquip[k] := 1;
            end; {for k}
            Tgt[j].TS[s] := (1 + Tgt[j].COA) * Tgt[j].Echelon * ETS;
            Tgt[j].CS[s] := (1 + Tgt[j].COA) * Tgt[j].Echelon * ECS;
            Tgt[j].DS[s] := Tgt[j].TS[s] * Phase[i].DL[Tgt[j].UnitType,s];
            Tgt[j].DRS[s]:= Tgt[j].CS[s] - Tgt[j].DS[s];
            if (Tgt[j].DRS[s] < 0.0) then Tgt[j].DRS[s] := 0.0;
          end; {if}
        end; {for j}
      end; {for s}
  end; {for i}
  for i := 1 to NumPhases do begin
      Str(Day.Num+1,DayCh);
      Str(i,PhaseCh);
      assign  (DRSfile,DataDir+'DRS'+Daych+PhaseCh+'.dat');
      assign  (ECNfile,DataDir+'ECN'+DayCh+PhaseCh+'.dat');
```

```
 assign  (PKfile,DataDir+'PK'+PhaseCh+'.dat');
 assign  (ESVfile,DataDir+'ESV'+PhaseCh+'.dat');
 rewrite (DRSfile);
 rewrite (ECNfile);
 rewrite (PKfile);
 rewrite (ESVfile);
 write (DRSfile,' ');
 write (ESVfile,' ');
 for s:= 1 to NumStrengths do begin
   if (PHase[i].ActiveStrength[s] = 1) then begin
     write (DRSfile,'     S',s);
     write (ESVfile, '     S',s);
   end; {if}
 end; {for s}
 writeln (DRSfile);
 writeln (ESVfile);
 write  (ECNfile,'  ');
 write  (PKfile, ' ');
 for k:= 1 to NumEquip do begin
   if (Phase[i].ActiveEquip[k] = 1) then
     if (k < 10) then begin
       write (ECNfile,'    E'+'0',k);
       write (PKfile,  '    E'+'0',k);
     end {if}
     else begin
       write (ECNfile,'    E',k);
       write (PKfile,  '    E',k);
     end; {else}
 end; {for k}
 writeln (ECNfile);
 writeln (PKfile);
 for j := 1 to NumTgts do begin
   write (DRSfile,'P',j);
   write (ECNfile,'P',j);
   for s := 1 to NumStrengths do begin
     if (Phase[i].ActiveStrength[s]=1) then write (DRSfile,Tgt[j].DRS[s]:8:1);
   end; {for s}
   writeln (DRSfile);
   for k := 1 to NumEquip do begin
     if (Phase[i].ActiveEquip[k]=1) then write (ECNfile, Tgt[j].CNE[k]:8);
   end; {for k}
   writeln (ECNfile);
 end; {for j}
```

```
        close (DRSfile);
        close (ECNfile);
        for j:=1 to NumBlueAC do begin
          write (PKfile,'A',j);
          for k:= 1 to NumEquip do begin
            if (Phase[i].ActiveEquip[k]=1) then write (PKfile,PKASD[j,k]:8:4);
          end; {for k}
          writeln (PKfile);
        end; {for j}
        close (PKfile);
        for k:=1 to NumEquip do begin
          if (Phase[i].ActiveEquip[k]=1) then begin
            if (k < 10) then write (ESVfile,'E'+'0',k)
            else write (ESVfile,'E',k);
            for s:=1 to NumStrengths do begin
              if (Phase[i].ActiveStrength[s]=1) then write (ESVfile,Equip[k,s]:8:1);
            end; {for s}
            writeln (ESVfile);
          end; {if}
        end; {for k}
        close (ESVfile);
    end; {for i}
    writeln ('** Target strengths computed and files created.');
end; {ComputeStrengths}
{** Procedure PhaseTransition
*********************************************}
procedure PhaseTransition;
var     i,j,k,s,n          : integer;
        ActivatePhase : Boolean;
        DayCh         : string;
        DayResults    : text;
begin
  for i := 1 to NumPhases do begin
    for s := 1 to NumStrengths do begin
      Phase[i].TS[s]      := 0.0;
      Phase[i].DS[s]      := 0.0;
      Phase[i].CS[s]      := 0.0;
      Phase[i].DRS[s]     := 0.0;
    end; {for s}
  end; {for i}
  for i := 1 to NumPhases do begin
    for s := 1 to NumStrengths do begin
      if (Phase[i].ActiveStrength[s] = 1) then begin
```

```
        for j := 1 to NumTgts do begin
          Phase[i].DRS[s] := Phase[i].DRS[s] + Tgt[j].DRS[s];
          Phase[i].TS[s]  := Phase[i].TS[s]  + Tgt[j].TS[s];
          Phase[i].DS[s]  := Phase[i].DS[s]  + Tgt[j].DS[s];
          Phase[i].CS[s]  := Phase[i].CS[s]  + Tgt[j].CS[s];
        end; {for j}
      end; {if}
    end; {for s}
  end; {for i}
  Str(Day.Num,DayCh);
  assign (DayResults,DataDir+'DayRes.dat');
  append (DayResults);
  writeln (DayResults);
        for i := 1 to NumPhases do begin
          ActivatePhase := True;
          if (Phase[i].Active = 1) then begin
            for s := 1 to NumStrengths do begin
              if (Phase[i].ActiveStrength[s] = 1) then begin
                if ((Phase[i].TS[s] > Phase[i].DS[s]) and
                   (Phase[i].DRS[s] /
                   (Phase[i].TS[s] - Phase[i].DS[s]) > Day.CriteriaVal)) then
                  ActivatePhase := false;
              end; {if}
            end; {for s}
            if (ActivatePhase) then begin
              Phase[i+1].Active := 1;
              writeln ('**  Phase ',i+1,' has been activated.');
              writeln ('    Criteria Value: ',Day.CriteriaVal:4);
              writeln (DayResults,'**  Phase ',i+1,' has been activated.');
              writeln (DAyResults,'    Criteria Value: ',Day.CriteriaVal:4);
            end; {if}
          end; {if}
        end; {for i}
  writeln (DayResults,'*************************************');
  writeln (DayResults,'      Strength Statistics');
  writeln (DayResults,'*************************************');
  for i := 1 to NumPhases do begin
    writeln (DayResults,' Phase ',I,' Strengths');
    writeln (DayResults,'  TS strengths');
    for s := 1 to NumStrengths do begin
      if (Phase[i].ActiveStrength[s] = 1) then
        write (DayResults,StrengthName[s]:7);
    end; {for s}
```

```
writeln (DayResults);
for j := 1 to NumTgts do begin
  write (DayResults,Tgt[j].ID:12);
  for s := 1 to NumStrengths do begin
    if (Phase[i].ActiveStrength[s] = 1) then begin
      if (Tgt[j].TS[s] > 0.0) then
        write (DayResults,Tgt[j].TS[s]:7:1)
      else write  (DayResults,'--':7);
    end; {if}
  end; {for s}
  writeln (DayResults);
end; {for j}
write (Dayresults,'  Total     ');
for s := 1 to NumStrengths do begin
  if (Phase[i].ActiveStrength[s] = 1) then
    write (DayResults, Phase[i].TS[s]:7:1);
end; {for s}
writeln (DayResults);
writeln (DayResults,'  DS strengths');
for s := 1 to NumStrengths do begin
  if (Phase[i].ActiveStrength[s] = 1) then
    write (DayResults,StrengthName[s]:7);
end; {for s}
writeln (DayResults);
for j := 1 to NumTgts do begin
  write (DayResults,Tgt[j].ID:12);
  for s := 1 to NumStrengths do begin
    if (Phase[i].ActiveStrength[s] = 1) then begin
      if (Tgt[j].DS[s] > 0.0) then
        write (DayResults,Tgt[j].DS[s]:7:1)
      else write  (DayResults,'--':7);
    end; {if}
  end; {for s}
  writeln (DayResults);
end; {for j}
write (Dayresults,'  Total     ');
for s := 1 to NumStrengths do begin
  if (Phase[i].ActiveStrength[s] = 1) then
    write (DayResults, Phase[i].DS[s]:7:1);
end; {for s}
writeln (DayResults);
writeln (DayResults,'  CS strengths');
for s := 1 to NumStrengths do begin
```

```
    if (Phase[i].ActiveStrength[s] = 1) then
      write (DayResults,StrengthName[s]:7);
  end; {for s}
  writeln (DayResults);
  for j := 1 to NumTgts do begin
    write (DayResults,Tgt[j].ID:12);
    for s := 1 to NumStrengths do begin
      if (Phase[i].ActiveStrength[s] = 1) then begin
        if (Tgt[j].DS[s] > 0.0) then
          write (DayResults,Tgt[j].CS[s]:7:1)
        else write  (DayResults,'--':7);
      end; {if}
    end; {for s}
    writeln (DayResults);
  end; {for j}
  write (Dayresults,'  Total     ');
  for s := 1 to NumStrengths do begin
    if (Phase[i].ActiveStrength[s] = 1) then
      write (DayResults, Phase[i].CS[s]:7:1);
  end; {for s}
  writeln (DayResults);
  writeln (DayResults,'   DRS strengths');
  for s := 1 to NumStrengths do begin
    if (Phase[i].ActiveStrength[s] = 1) then
      write (DayResults,StrengthName[s]:7);
  end; {for s}
  writeln (DayResults);
  for j := 1 to NumTgts do begin
    write (DayResults,Tgt[j].ID:12);
    for s := 1 to NumStrengths do begin
      if (Phase[i].ActiveStrength[s] = 1) then begin
        if (Tgt[j].DRS[s] > 0.0) then
          write (DayResults,Tgt[j].DRS[s]:7:1)
        else write  (DayResults,'--':7);
      end; {if}
    end; {for s}
    writeln (DayResults);
  end; {for j}
  write (Dayresults,'  Total     ');
  for s := 1 to NumStrengths do begin
    if (Phase[i].ActiveStrength[s] = 1) then
      write (DayResults, Phase[i].DRS[s]:7:1);
  end; {for s}
```

```
      writeln (DayResults);
      writeln (DayResults,'  % of strength left to destroy');
      for s := 1 to NumStrengths do begin
        if (Phase[i].ActiveStrength[s] = 1) then
          write (DayResults,StrengthName[s]:7);
      end; {for s}
      writeln (DayResults);
      for j := 1 to NumTgts do begin
        write (DayResults,Tgt[j].ID:12);
        for s := 1 to NumStrengths do begin
          if (Phase[i].ActiveStrength[s] = 1) then begin
            if (Tgt[j].TS[s] > 0.0) then
              if (Tgt[j].CS[s] > Tgt[j].DS[s]) then
                write (DayResults,(Tgt[j].DRS[s] /
                      (Tgt[j].TS[s] - Tgt[j].DS[s]))*100:7:1)
              else
                write (DayResults,'0':7)
            else write  (DayResults,'--':7);
          end; {if}
        end; {for s}
        writeln (DayResults);
      end; {for j}
      write (Dayresults,'  Ave. %   ');
      for s := 1 to NumStrengths do begin
        if (Phase[i].ActiveStrength[s] = 1) then begin
          if (Phase[i].TS[s] > Phase[i].DS[s]) then
            write (DayResults, (Phase[i].DRS[s] /
                  (Phase[i].TS[s] - Phase[i].DS[s]))*100:7:1)
          else
            write (DayResults,'0':7);
        end; {if}
      end; {for s}
      writeln (DayResults);
      writeln (DayResults);
    end; {for i}
    close (DayResults);
  end; {Phasetransition}
end. {unit Strengths}
```

F. ENEMY AIR DEFENSE GENERATION MODULE

```
{*************************************************************************
**
   Module:      BuildEAD.PAS
   Description: Module builds enemy air defense network based on current
                number of surface-to-air and air-to-air systems.  Also
                creates PSAA and PSSA files for air allocation program.
*************************************************************************
*}
Unit BuildEADU;
interface
uses Data,In_Out,PKTool1,PKTool2,MRoutool;
Procedure BuildEAD (var Day: Daytype;var Air_Grid: Grid_Value;
                    var Tgt:Tgtdata; var BlueAB: BlueABdata);
implementation
Procedure BuildEAD;
var     h,i,j,k,l,m,n,number          : integer;
        delta,Range,PKill             : real;
        TotArea,PK,Store,PKAA,PKSA    : Keep_Value;
        Totally_inside,Inside         : boolean;
        PSSAfile,PSAAfile,PKEADfile,
  RANGEfile                   : text;
  DayChar                     : string;
  g                           : VertexList;
begin
  Delta := 0.1;
  Str(Day.Num+1,DayChar);
  assign  (PSSAfile,DataDir+'PSSA'+DayChar+'.dat');
  assign  (PSAAfile,DataDir+'PSAA'+DayChar+'.dat');
  assign  (RANGEfile,DataDir+'RANGE'+Daychar+'.dat');
  assign  (PKEADfile,DataDir+'PKEAD'+DayChar+'.dat');
  rewrite (PSSAfile);
  rewrite (PSAAfile);
  rewrite (RANGEfile);
  rewrite (PKEADfile);
  writeln (PSSAfile,'          ES0       ES1');
  writeln (PSAAfile,'          EF0       EF1');
  write   (RANGEfile,'     ');
  for i := 1 to NumTgts do begin
    write (RANGEfile,' T',i);
  end; {for i}
```

```
writeln (RANGEfile);
writeln ('** Building Enemy Air Defense data.');
for l := 1 to NumBlueAB do begin
  for m := 1 to NumBlueAC do begin
    if (BlueAB[l].CNE[m] > 0) then begin
      Initial_state (Day,PKAA,TotArea);
      Initial_state (Day,PK,TotArea);
      for i := 1 to NumTgts do begin
        Inside_or_not(Day,inside,number,Tgt[i].GridX,Tgt[i].GridY);
        for j := 1 to NumEquip do begin
          if (j = 4) then begin
            for k := 1 to Tgt[i].CNE[j] do begin
              Initial_State (Day,PK,store);
              {AA PK information}
              Range := PKAAD[1];
              PKill := PKAAD[1+m];
              Caculation_of_PK_and_difficulty_level
               (Day,inside,totally_inside,Tgt[i].GridX,Tgt[i].GridY,
                Range,PKill,delta,number,air_grid,PK,store);
              Area_calculation_of_special_case
               (Day,inside,totally_inside,air_grid,number,Tgt[i].GridX,
                Tgt[i].GridY,Range,PKill,delta,PK,Store);
              Keep(Day,PKAA,TotArea,PK,Store);
              write ('.');
            end; {for k}
          end; {if}
        end; {for j}
      end; {for i}
      Initial_state (Day,PKSA,TotArea);
      for i := 1 to NumTgts do begin
        Inside_or_not(Day,inside,number,Tgt[i].GridX,Tgt[i].GridY);
        for j := 1 to NumEquip do begin
          if (j = 5) or (j = 6) then begin
            for k := 1 to Tgt[i].CNE[j] do begin
              {SAM PK information}
              Initial_state (Day,PK,store);
              if (j= 5) then begin
                Range := PKSAD[1,1];
                PKill := PKSAD[1,1+m];
              end {if}
              else begin
                Range := PKSAD[2,1];
                PKill := PKSAD[2,1+m];
```

```
                end; {else}
                Caculation_of_PK_and_difficulty_level
                  (Day,inside,totally_inside,Tgt[i].GridX,Tgt[i].GridY,
                   Range,PKill,delta,number,air_grid,PK,store);
                Area_calculation_of_special_case
                  (Day,inside,totally_inside,air_grid,number,Tgt[i].GridX,
                   Tgt[i].GridY,Range,PKill,delta,PK,Store);
                Keep(Day,PKSA,TotArea,PK,Store);
                write ('.');
              end; {for k}
            end; {if}
          end; {for j}
        end; {for i}
        NetworkInput (g);
        Search_Part  (g,BlueAB[l].Grid,PKSA,PKAA);
        write  (RANGEfile,'B',l,'.A',m,'  ');
        for i := 1 to NumTgts do begin
          writeln (PSSAfile,'B',l,'.A',m,'.T',i,'    '
                   ,(1-g[Tgt[i].Grid].PKSA):8:4,'    '
                   ,(1-0.25*g[Tgt[i].Grid].PKSA):8:4);
          writeln (PSAAfile,'B',l,'.A',m,'.T',i,'    '
                   ,(1-g[Tgt[i].Grid].PKAA):8:4,'    '
                   ,(1-0.25*g[Tgt[i].Grid].PKAA):8:4);
          if (g[Tgt[i].Grid].RouteDis*GridWidth <= BlueAC[m,1]) then
            write (RANGEfile,' 1 ')
          else write (RANGEfile,'  ');
        end; {for i}
        writeln (RANGEfile);
        writeln
(PKEADfile,'*************************************************');
        writeln (PKEADfile,'Air Base: ',BlueAB[l].ID,' Grid Loc: ',BlueAB[l].Grid,'
Aircraft: ',
                 BlueAB[l].SquadID[m]);
        writeln
(PKEADfile,'*************************************************');
        writeln (PKEADfile,'Tgt  Grid PSSA   PSAA    DIST    Route');
        for i := 1 to NumTgts do begin
          write (PKEADfile,i,Tgt[i].Grid:7,(1-g[Tgt[i].Grid].PKSA):7:4,
                   (1-g[Tgt[i].Grid].PKAA):7:4,
                  (g[Tgt[i].Grid].RouteDis*GridWidth):8:1,'  ');
          n := Tgt[i].Grid;
          if (Tgt[i].Grid <> BlueAB[l].Grid) then
            while (n <> BlueAB[l].Grid) do begin
```

```
                n := g[n].next_choice;
                write (PKEADfile,n,'-');
              end; {while}
            writeln (PKEADfile);
          end; {for}
        end; {if}
      end; {for m}
    end; {for l}
    writeln;
    writeln ('**  Enemy Air Defense data computed and stored.');
    writeln ('**  All files created and saved.  Ready for GAMS day ',DayChar);
    close  (PSSAfile);
    close  (PSAAfile);
    close  (RANGEfile);
    close  (PKEADfile);
  end; {BuildAA}
end. {unit BuildEAD}
```

APPENDIX C. SAMPLE SORTIE ALLOCATION MIP OUTPUT

The following is a sample output listing of the sortie allocation MIP output. This output was for the first day and first phase of the example campaign. Lines 283 through 286 of the output give the number of AI, fighter, and SEAD sorties allocated.

```
GAMS 2.25.060  386/486 DOS                    09/10/95 14:49:20  PAGE    1
NAME = JEFFREY P. HAMMAN
THESIS AIR ALLOCATION PROGRAM
  3 *---GAMS AND DOLLAR CONTROL OPTIONS----------------------------------*
  5
  6 OPTIONS
  7  LIMCOL = 0,   LIMROW = 0,   SOLPRINT = OFF, DECIMALS = 2
  8  RESLIM = 550, ITERLIM = 5000, OPTCR   = 0.1, SEED    = 3141;
  9 *---------------------------------------------------------------------*
 10
 11 SETS
 12       A  aircraft types          / A1 * A4 /
 13       B  air base ID             / B1 * B2 /
 14       P  target ID               / P1 * P8 /
 15       S  strength category       / S1,S3,S5,S6 /
 16       E  equipment types         / E03,E04,E05,E06,E07,E08 /
 17       EF escort fighter          / EF0,EF1 /
 18       ES escort SEAD             / ES0,ES1 /
 19       ;
 20
 21 SCALAR  W  Maximum attrition fraction /0.15/;
 22
 23 TABLE PK(A,E) Probability of kill kill of equipment type E by aircraft type A
INCLUDE    D:\THESIS\DATA\PK1.DAT
 25       E03     E04     E05     E06     E07     E08
 26 A1  0.4410  0.4410  0.3780  0.2250  0.3150  0.3150
 27 A2  0.4730  0.4730  0.4050  0.2250  0.3380  0.3380
 28 A3  0.4320  0.4320  0.4320  0.2430  0.3240  0.3600
 29 A4  0.4730  0.4730  0.4050  0.1130  0.3380  0.3380
 30 ;
 31
 32 TABLE DRS(P,S) Potential target desired reduction in strength.
INCLUDE    D:\THESIS\DATA\DRS11.DAT
 34       S1      S3      S5      S6
 35 P1   351.0  37.8   10.5   19.2
```

```
 36 P2    0.0   54.0   10.5    0.0
 37 P3    0.0   54.0   10.5    0.0
 38 P4    0.0   27.0    5.3    0.0
 39 P5    0.0   27.0    5.3    0.0
 40 P6    0.0   27.0   15.8    0.0
 41 P7    0.0   27.0   15.8    0.0
 42 P8    0.0   18.9    0.0    0.0
 43 ;
 44
 45 TABLE ECN(P,E)  Potential target current number of equipment.
INCLUDE   D:\THESIS\DATA\ECN11.DAT
 47        E03   E04   E05   E06   E07   E08
 48 P1      20    20     4     0     4     8
 49 P2       0     0     0     8     4     0
 50 P3       0     0     0     8     4     0
 51 P4       0     0     0     4     2     0
 52 P5       0     0     0     4     2     0
 53 P6       0     0     0     4     6     0
 54 P7       0     0     0     4     6     0
 55 P8       0     0     2     0     0     0
 56 ;
 57
 58 TABLE ESV(E,S)  Equipment strength value.
INCLUDE   D:\THESIS\DATA\ESV1.DAT
 60          S1    S3    S5    S6
 61 E03     2.0   0.0   0.0   0.0
 62 E04    11.0   0.0   0.0   0.0
 63 E05     0.0   7.0   0.0   0.0
 64 E06     0.0   5.0   0.0   0.0
 65 E07     0.0   0.0   2.5   0.0
 66 E08     0.0   0.0   0.0   2.0
 67 ;
 68
 69 TABLE RANGE(B,A,P) In range matrix for potential targets and aircraft types.
INCLUDE   D:\THESIS\DATA\RANGE1.DAT
 71            P1 P2 P3 P4 P5 P6 P7 P8
 72 B1.A2    1  1  1  1  1     1
 73 B1.A4    1  1  1  1  1     1
 74 B2.A1    1  1  1  1  1  1  1  1
 75 B2.A3    1  1  1  1  1  1
 76 ;
 77
```

78 TABLE MAXSORTIES(B,A) Maximum sorties available by air base and aircraft type.

```
INCLUDE   D:\THESIS\DATA\AB11.DAT
```

```
80        A1   A2   A3   A4
81 B1      0   120    0   40
82 B2     60    0   120    0
83 ;
84
```

85 TABLE PSAA(B,A,P,EF) Probability of successfully flying from air base to potential target due to enemy air-to-air defenses for aircraft type with or without escort fighter package.

```
INCLUDE   D:\THESIS\DATA\PSAA1.DAT
```

```
87                    EF0       EF1
88  B1.A2.P1        0.1506    0.7877
89  B1.A2.P2        0.1506    0.7876
90  B1.A2.P3        0.0959    0.7740
91  B1.A2.P4        0.1107    0.7777
92  B1.A2.P5        0.0506    0.7626
93  B1.A2.P6        0.0744    0.7686
94  B1.A2.P7        0.0102    0.7526
95  B1.A2.P8        0.0257    0.7564
96  B1.A4.P1        0.9778    0.9944
97  B1.A4.P2        0.9778    0.9944
98  B1.A4.P3        0.9725    0.9931
99  B1.A4.P4        0.9728    0.9932
100 B1.A4.P5        0.9652    0.9913
101 B1.A4.P6        0.9682    0.9921
102 B1.A4.P7        0.9633    0.9908
103 B1.A4.P8        0.9655    0.9914
104 B2.A1.P1        0.0169    0.7542
105 B2.A1.P2        0.0127    0.7532
106 B2.A1.P3        0.1156    0.7789
107 B2.A1.P4        0.0068    0.7517
108 B2.A1.P5        0.2157    0.8039
109 B2.A1.P6        0.0113    0.7528
110 B2.A1.P7        0.2281    0.8070
111 B2.A1.P8        0.0726    0.7681
112 B2.A3.P1        0.1341    0.7835
113 B2.A3.P2        0.1162    0.7791
114 B2.A3.P3        0.3452    0.8363
115 B2.A3.P4        0.0855    0.7714
116 B2.A3.P5        0.4695    0.8674
117 B2.A3.P6        0.1090    0.7773
```

```
 118 B2.A3.P7     0.4824     0.8706
 119 B2.A3.P8     0.2737     0.8184
 120 ;
 121
 122 TABLE  PSSA(B,A,P,ES) Probability of successfully flying from air base to
potential target due to enemy surface-to-air defenses for aircraft type with or without
escort SEAD package.
INCLUDE    D:\THESIS\DATA\PSSA1.DAT
 124                  ES0        ES1
 125 B1.A2.P1     0.3769     0.8442
 126 B1.A2.P2     0.5269     0.8817
 127 B1.A2.P3     0.3475     0.8369
 128 B1.A2.P4     0.7456     0.9364
 129 B1.A2.P5     0.2467     0.8117
 130 B1.A2.P6     0.8466     0.9616
 131 B1.A2.P7     0.6071     0.9018
 132 B1.A2.P8     0.5653     0.8913
 133 B1.A4.P1     0.9907     0.9977
 134 B1.A4.P2     0.9939     0.9985
 135 B1.A4.P3     0.9898     0.9974
 136 B1.A4.P4     0.9925     0.9981
 137 B1.A4.P5     0.9866     0.9966
 138 B1.A4.P6     0.9938     0.9984
 139 B1.A4.P7     0.9838     0.9960
 140 B1.A4.P8     0.9866     0.9966
 141 B2.A1.P1     0.1949     0.7987
 142 B2.A1.P2     0.1720     0.7930
 143 B2.A1.P3     0.5515     0.8879
 144 B2.A1.P4     0.4221     0.8555
 145 B2.A1.P5     0.6903     0.9226
 146 B2.A1.P6     0.6575     0.9144
 147 B2.A1.P7     0.8097     0.9524
 148 B2.A1.P8     0.6011     0 9003
 149 B2.A3.P1     0.2747     0.8187
 150 B2.A3.P2     0.2486     0.8121
 151 B2.A3.P3     0.6247     0.9062
 152 B2.A3.P4     0.5046     0.8761
 153 B2.A3.P5     0.7456     0.9364
 154 B2.A3.P6     0.7172     0.9293
 155 B2.A3.P7     0.8466     0.9616
 156 B2.A3.P8     0.6678     0.9170
 157 ;
 158
```

```
159 PARAMETER COA(P)  Course of action probabilities for potential targets.
INCLUDE   D:\THESIS\DATA\COA1.DAT
161 / P1    0.6
162  P2     0.4
163  P3     0.6
164  P4     0.4
165  P5     0.6
166  P6     0.4
167  P7     0.6
168  P8     0.4
169 /
170 ;
171
172 PARAMETER ECHELON(P)  Echelon weights for potential targets.
INCLUDE   D:\THESIS\DATA\ECH1.DAT
174 / P1    1.0
175  P2     1.0
176  P3     1.0
177  P4     1.0
178  P5     1.0
179  P6     1.0
180  P7     1.0
181  P8     1.0
182 /
183 ;
184
185 PARAMETER MAXSEAD(B) Maximum number of SEAD packages available
from each air base.
INCLUDE   D:\THESIS\DATA\MAXSEA11.DAT
187 /B1    12
188  B2    12/
189 ;
190
191 PARAMETER MAXFIGHT(B) Maximum number of fighter packages available
from each air base.
INCLUDE   D:\THESIS\DATA\MAXFTR11.DAT
193 /B1    20
194  B2    20/
195 ;
196
197
198 VARIABLES
199
```

```
200   VAL                          Objective value.
201   TSD(P,S)                     Total target strength value destroyed.
202   SORTIES(B,A,P,S,E,EF,ES)     Number of sorties allocated to potential target P,
                                   attacking equipment E, with or without escort
                                   fighter/SEAD packages.
203   EA(B,A,P,S,E,EF,ES)          Expected attrition of flight group flying to potential
                                   target P.
204   SEAD(B,A,P)                  Number of escort SEAD packages assigned to
                                   escort aircraft A flying to potential target P.
205   FIGHT(B,A,P)                 Number of escort fighter packages assigned to
                                   escort aircraft A flying to potential target P.
206
207   POSITIVE VARIABLE SORTIES;
208   POSITIVE VARIABLE TSD,EA;
209   INTEGER  VARIABLE FIGHT,SEAD;
210   SORTIES.UP(B,A,P,S,E,EF,ES) = MAXSORTIES(B,A);
211   SORTIES.LO(B,A,P,S,E,EF,ES) = 0;
212   FIGHT.LO(B,A,P) = 0;
213   FIGHT.UP(B,A,P) = MAXFIGHT(B);
214   FIGHT.LO(B,A,P) = 0;
215   SEAD.UP(B,A,P)  = MAXSEAD(B);
216
217 EQUATIONS
218   VALUE                        Objective function
219   EATT(B,A,P)                  Attrition limit
220   TSDEQ(P,S)                        Strength category value destroyed
221   MAXSORTC(B,A)                Available aircraft by type
222   TSDVALC(P,S)                 Destroy no more than whats there
223   EQKILLED(P,E)                Destroy no more than whats there
224   EAEQ(B,A,P,S,E,EF,ES)        Expected attrition
225   SEADPACK(B,A,P)              Available SEAD packages
226   FIGHTPACK(B,A,P)             Available fighter packages
227   MAXSEADEQ(B)                 Total escort SEAD packages
228   MAXFIGHTEQ(B)                Total escort fighter packages;
229
230 VALUE..    VAL =E= SUM((P,S),TSD(P,S));
231
232 EATT(B,A,P).. SUM((S,E,EF,ES),EA(B,A,P,S,E,EF,ES) -
                          W*,SORTIES(B,A,P,S,E,EF,ES));
234
235 TSDEQ(P,S)..
236     TSD(P,S) =E=
    (1+COA(T))*ECHELON(T)*SUM((A,B,E),PK(A,E)*ESV(E,S)*
```

```
237         (SUM((EF,ES),RANGE(B,A,P)*SORTIES(B,A,P,S,E,EF,ES)
238                 -EA(B,A,P,S,E,EF,ES))));
239
240 MAXSORTC(B,A)..
241     SUM((P,S,E,EF,ES),SORTIES(B,A,P,S,E,EF,ES)) =L= MAXSORTIES(B,A);
242
243 TSDVALC(P,S)..
244     TSD(P,S) =L= DRS(P,S);
245
246 EQKILLED(P,E)..
247   SUM((B,A,S),SUM((EF,ES),SORTIES(B,A,P,S,E,EF,ES)-EA(B,A,P,S,EF,ES))
248               *PK(A,E))=L= ECN(P,E);
249
250 EAEQ(B,A,P,S,E,EF,ES)..
251     EA(B,A,P,S,E,EF,ES) =E= SORTIES(B,A,P,S,E,EF,ES)*
252                         (1-(PSAA(B,A,P,EF)*PSSA(B,A,P,ES)));
253
254 SEADPACK(B,A,P)..
255     SUM((S,E,EF),SORTIES(B,A,P,S,E,EF,'ES1')) =L=
            SEAD(B,A,P)*MAXSEADESC;
256
257 FIGHTPACK(B,A,P)..
258     SUM((S,E,ES),SORTIES(B,A,P,S,E,'EF1',ES)) =L=
            FIGHT(B,A,P)*MAXFTRESC;
259
260 MAXSEADEQ(B)..
261     SUM((A,P),SEAD(B,A,P)) =L= MAXSEAD(B);
262
263 MAXFIGHTEQ(B)..
264     SUM((A,P),FIGHT(B,A,P)) =L= MAXFIGHT(B);
265
266
267 MODEL ALLOC /ALL/;
268 SOLVE ALLOC USING MIP MAXIMIZING VAL;
269
270 PARAMETER MISSIONS   (*,*,*,*,*,*);
271           MISSIONS(B,A,P,E,EF,ES) = SUM(S,SORTIES.L(B,A,P,S,E,EF,ES));
272 PARAMETER MISSIONEA  (*,*,*,*,*,*);
273           MISSIONEA(B,A,P,E,EF,ES) = SUM(S,EA.L(B,A,P,S,E,EF,ES));
274 PARAMETER SORTIESUSE (*,*);
275           SORTIESUSE(B,A) =
                    SUM((P,S,E,EF,ES),SORTIES.L(B,A,P,S,E,EF,ES));
276 PARAMETER SORTIEAVAL (*,*);
```

```
 277          SORTIEAVAL(B,A) = MAXSORTIES(B,A)
                                SUM((P,S,E,EF,ES),SORTIES.L(B,A,P,S,E,EF,ES));
 278  PARAMETER FIGHTAVAIL (*);
 279          FIGHTAVAIL(B) = MAXFIGHT(B) - SUM((A,P),FIGHT.L(B,A,P));
 280  PARAMETER SEADAVAIL (*);
 281          SEADAVAIL(B)  = MAXSEAD(B)  - SUM((A,P),SEAD.L(B,A,P));
 282
 283  OPTION MISSIONS:0:0:1;    DISPLAY MISSIONS;
 284  OPTION FIGHT:0:0:1;       DISPLAY FIGHT.L;
 285  OPTION SEAD:0:0:1;        DISPLAY SEAD.L;
 286  OPTION SORTIEAVAL:0:0:1 DISPLAY SORTIEAVAL;
 287  OPTION FIGHTAVAIL:0:0:1  DISPLAY FIGHTAVAIL;
 288  OPTION SEADAVAIL:0:0:1   DISPLAY SEADAVAIL;
 289  OPTION MISSIONEA:0:0:1   DISPLAY MISSIONEA;
 290  DISPLAY SORTIESUSE;
 291  OPTION EA:1:0:1;          DISPLAY EA.L;
 292  DISPLAY TDRS,TSD.L;
GAMS 2.25.060  386/486 DOS                09/10/95 14:49:20  PAGE     2
NAME = JEFFREY P. HAMMAN
Include File Summary
  GLOBAL TYPE       LOCAL  FILE NAME
     0 INPUT          0  D:\THESIS\GAMS\AIRALL11.GMS
    24 INCLUDE       24  .D:\THESIS\DATA\EK1.DAT
    33 INCLUDE       28  .D:\THESIS\DATA\TDRS11.DAT
    46 INCLUDE       32  .D:\THESIS\DATA\TECN11.DAT
    59 INCLUDE       36  .D:\THESIS\DATA\ESV1.DAT
    70 INCLUDE       40  .D:\THESIS\DATA\RANGE1.DAT
    79 INCLUDE       44  .D:\THESIS\DATA\AB11.DAT
    86 INCLUDE       48  .D:\THESIS\DATA\PSAA1.DAT
   123 INCLUDE       52  .D:\THESIS\DATA\PSSA1.DAT
   160 INCLUDE       56  .D:\THESIS\DATA\COA1.DAT
   173 INCLUDE       60  .D:\THESIS\DATA\ECH1.DAT
   186 INCLUDE       64  .D:\THESIS\DATA\MAXSEA11.DAT
   192 INCLUDE       68  .D:\THESIS\DATA\MAXFTR11.DAT
COMPILATION TIME     =        2.140 SECONDS      VERID MW2-00-060
GAMS 2.25.060  386/486 DOS                09/10/95 14:49:20  PAGE     3
NAME = JEFFREY P. HAMMAN
Model Statistics    SOLVE ALLOC USING MIP FROM LINE 268
MODEL STATISTICS
BLOCKS OF EQUATIONS     11    SINGLE EQUATIONS     6461
BLOCKS OF VARIABLES      6    SINGLE VARIABLES    12449
NON ZERO ELEMENTS    51665    DISCRETE VARIABLES   3200
GENERATION TIME      =       26.250 SECONDS
```

```
EXECUTION TIME      =      27.130 SECONDS      VERID MW2-00-060
GAMS 2.25.060  386/486 DOS                    09/10/95 14:49:20  PAGE     4
NAME = JEFFREY P. HAMMAN
Solution Report     SOLVE ALLOC USING MIP FROM LINE 268
          S O L V E     S U M M A R Y
    MODEL   ALLOC            OBJECTIVE  VAL
    TYPE    MIP             DIRECTION  MAXIMIZE
    SOLVER  XA               FROM LINE  268
**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      8 INTEGER SOLUTION
**** OBJECTIVE VALUE              330.3657
 RESOURCE USAGE, LIMIT         15.000     550.000
 ITERATION COUNT, LIMIT       127        5000
 ************************************************************
 *  XA Linear & Generalized Integer Program
 *     Copyright 1991, 1992, 1993 by Sunset Software Technology
 *     All Rights Reserved Worldwide.
 *     Phone 818-441-1565   FAX 818-441-1567
 ************************************************************
Note 2. The stopping tolerance is satisfied,
        but the solution may not be optimal.
No better solution than :   330.36565
                    Absolute            Relative
Actual distance         0.00000             0.00000
Tolerances  (OPTCA)     0.00000  (OPTCR)    0.10000
*** End of XA Messages ******************************
**** REPORT SUMMARY :        0     NONOPT
                            0 INFEASIBLE
                            0  UNBOUNDED
GAMS 2.25.060  386/486 DOS                    09/10/95 14:49:20  PAGE     5
NAME = JEFFREY P. HAMMAN
E x e c u t i o n
----    283 PARAMETER MISSIONS
B1.A4.P1.E04.EF1.ES1 40
----    284 VARIABLE  FIGHT.L       Escort Fighter package.
B1.A4.P1 7
----    285 VARIABLE  SEAD.L        Escort SEAD package.
B1.A4.P1 7
----    286 PARAMETER SORTIEAVAL
B1.A2 120
B2.A1  60
B2.A3 120
----    287 PARAMETER FIGHTAVAIL
```

```
B1 13
B2 20
----    288 PARAMETER SEADAVAIL
B1  5
B2 12
----    289 PARAMETER MISSIONEA
B1.A4.P1.E04.EF1.ES1 3.154848E-1
----    290 PARAMETER SORTIESUSE
         A4
B1      40.00
----    291 VARIABLE  EA.L
B1.A4.P1.S1.E04.EF1.ES1 0.3
----    292 PARAMETER DRS
         S1        S3        S5        S6
P1     351.00  37.80     10.50     19.20
P2               54.00     10.50
P3               54.00     10.50
P4               27.00      5.30
P5               27.00      5.30
P6               27.00     15.80
P7               27.00     15.80
P8               18.90
----    292 VARIABLE  TSD.L
         S1
P1     330.37
EXECUTION TIME       =        4.120 SECONDS       VERID MW2-00-060
USER: Operations Research Department              G931001-1228Ax-MW2
     Naval Postgraduate School
**** FILE SUMMARY
INPUT      D:\THESIS\GAMS\AIRALL11.GMS
OUTPUT     D:\THESIS\GAMS\AIRALL11.LST
```

APPENDIX D. MODEL DATA

The data provided in Appendix D were used for the demonstration of the $(AA)^2$ model.

1. AIRCRAFT AIR-TO-SURFACE PROBABILITY DATA

The data which follows are used to compute the probability of kill for a friendly aircraft type when attacking an equipment system. The resultant value, $P_{KILL|SHOT}$ for each aircraft, a, and equipment type, e, combination is assigned to the variable, $PKASD_{ae}$. These values were selected for the model demonstration and do not represent actual aircraft performance data.

A-6E					
EQUIP TYPE	P_{DET} x $P_{SHOT\|DET}$ x $P_{HIT\|SHOT}$ x $P_{KILL\|HIT}$ = $P_{KILL\|SHOT}$				
TANK	0.5	1.0	0.5	0.9	.225
ARTY	0.7	1.0	0.7	0.9	.441
ATTACK JET	0.7	1.0	0.7	0.9	.441
FIGHTER JET	0.7	1.0	0.7	0.9	.441
FIXED SAM	0.6	1.0	0.7	0.9	.378
MOBILE SAM	0.5	1.0	0.5	0.9	.225
C^2 VAN	0.5	1.0	0.7	0.9	.315
RUNWAY	0.5	1.0	0.7	0.9	.315
POL BLIVET	0.9	1.0	0.7	0.9	.567
POL TRUCK	0.7	1.0	0.7	0.9	.441

Table 11. A-6E aircraft air-to-surface probabilities.

F-15E					
EQUIP TYPE	**P_{DET} x $P_{SHOT\|DET}$ x $P_{HIT\|SHOT}$ x $P_{KILL\|HIT}$ = $P_{KILL\|SHOT}$**				
TANK	0.5	1.0	0.6	0.9	.270
ARTY	0.7	1.0	0.75	0.9	.473
ATTACK JET	0.7	1.0	0.75	0.9	.473
FIGHTER JET	0.7	1.0	0.75	0.9	.473
FIXED SAM	0.6	1.0	0.75	0.9	.405
MOBILE SAM	0.5	1.0	0.5	0.9	.225
C^2 VAN	0.5	1.0	0.75	0.9	.338
RUNWAY	0.5	1.0	0.75	0.9	.338
POL BLIVET	0.9	1.0	0.75	0.9	.608
POL TRUCK	0.7	1.0	0.75	0.9	.473

Table 12. F-15E aircraft air-to-surface probabilities.

F/A-18C					
EQUIP TYPE	**P_{DET} x $P_{SHOT\|DET}$ x $P_{HIT\|SHOT}$ x $P_{KILL\|HIT}$ = $P_{KILL\|SHOT}$**				
TANK	0.40	1.0	0.7	0.9	.216
ARTY	0.60	1.0	0.8	0.9	.432
ATTACK JET	0.60	1.0	0.8	0.9	.432
FIGHTER JET	0.60	1.0	0.8	0.9	.432
FIXED SAM	0.60	1.0	0.8	0.9	.432
MOBILE SAM	0.45	1.0	0.6	0.9	.243
C^2 VAN	0.45	1.0	0.8	0.9	.324
RUNWAY	0.50	1.0	0.8	0.9	.360
POL BLIVET	0.90	1.0	0.8	0.9	.648
POL TRUCK	0.65	1.0	0.8	0.9	.468

Table 13. F/A-18C aircraft air-to-surface probabilities.

F-117A					
EQUIP TYPE	**P_{DET}**	**$P_{SHOT\|DET}$**	**$P_{HIT\|SHOT}$**	**$P_{KILL\|HIT}$**	**$P_{KILL\|SHOT}$**
TANK	0.25	1.0	0.60	0.9	.135
ARTY	0.25	1.0	0.75	0.9	.168
ATTACK JET	0.70	1.0	0.75	0.9	.473
FIGHTER JET	0.70	1.0	0.75	0.9	.473
FIXED SAM	0.60	1.0	0.75	0.9	.405
MOBILE SAM	0.25	1.0	0.5	0.9	.113
C^2 VAN	0.50	1.0	0.75	0.9	.338
RUNWAY	0.50	1.0	0.75	0.9	.338
POL BLIVET	0.90	1.0	0.75	0.1	.068
POL TRUCK	0.70	1.0	0.75	0.9	.473

Table 14. F-117A aircraft air-to-surface probabilities.

2. ENEMY AIR DEFENSE PROBABILITIES

The following data were used to compute the probability of kill for enemy air defense systems against friendly air interdiction aircraft. The resultant values of $P_{KILL|SHOT}$ are assigned to the variables $PKAAD_{ae}$ and $PKSAD_{ae}$. These values were selected for model demonstration purposes only and do not represent actual system performance.

FIGHTER AIRCRAFT	P_{DET}	x $P_{SHOT\|DET}$	x $P_{HIT\|SHOT}$	x $P_{KILL\|HIT}$	= $P_{KILL\|SHOT}$
A-6E	0.200	0.7	0.5	0.9	0.06300
F-15E	0.100	0.7	0.5	0.9	0.03150
F-18C	0.100	0.7	0.5	0.9	0.03150
F-117A	0.001	0.7	0.5	0.9	0.00038

Table 15. Enemy fighter air-to-air probabilities.

FIXED SAM AIRCRAFT	P_{DET}	x $P_{SHOT\|DET}$	x $P_{HIT\|SHOT}$	x $P_{KILL\|HIT}$	= $P_{KILL\|SHOT}$
A-6E	02500	1.0	0.5	0.9	0.11250
F-15E	0.200	1.0	0.5	0.9	0.09000
F-18C	0.200	1.0	0.5	0.9	0.09000
F-117A	0.002	1.0	0.5	0.9	0.00090

Table 16. Enemy fixed SAM probabilities.

MOBILE SAM AIRCRAFT	P_{DET}	x $P_{SHOT\|DET}$	x $P_{HIT\|SHOT}$	x $P_{KILL\|HIT}$	= $P_{KILL\|SHOT}$
A-6E	0.300	1.0	0.6	0.8	0.11400
F-15E	0.250	1.0	0.6	0.8	0.12000
F-18C	0.250	1.0	0.6	0.8	0.12000
F-117A	0.004	1.0	0.6	0.8	0.00190

Table 17. Enemy mobile SAM probabilities.

3. POTENTIAL TARGET TO&E DATA

Air Base

20 - Fighter Aircraft
20 - Attack Aircraft
4 - Fixed SAMs
4 - C^2 Vans
8 - Runway Sections (16,000' Total)
10 - POL Blivets
10 - POL Trucks

Armor Brigade

80 - Tanks
8 - Mobile SAMs
4 - C^2 Vans

Artillery Battalion

18 - Artillery
4 - Mobile SAMs
2 - C^2 Vans

Logistic Unit (Mobile)

4 - Mobile SAMs
6 - C^2 Vans
10 - POL Blivets
20 - POL Trucks

Logistic Base (Fixed)

2 - Fixed SAMs
20 - POL Blivets
40 - POL Trucks

4. EQUIPMENT STRENGTH VALUES

The equipment strength values are used to value the different equipment types. These values are used in the calculations of strength category values for each unit. These

values are assigned to the variable ESV_{es} for each equipment type, e, and strength category, s.

	STRENGTH CATEGORIES							
EQUIP TYPE	**AAA**	**AAS**	**ASA**	**ASS**	**CC2**	**MOB**	**LPO**	**LST**
FIGHTER JET	11.0	4.5	0.0	0.0	0.0	0.0	0.0	0.0
ATTACK JET	2.0	12.0	0.0	0.0	0.0	0.0	0.0	0.0
FIXED SAM	0.0	0.0	7.0	0.0	0.0	0.0	0.0	0.0
MOBILE SAM	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0
TANK	0.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0
ARTILLERY	0.0	0.0	0.0	12.0	0.0	0.0	0.0	0.0
C2 VAN	0.0	0.0	0.0	0.0	2.5	0.0	0.0	0.0
RWY SEC	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0
POL BLIVET	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0
POL TRUCK	0.0	0.0	0.0	0.0	0.0	0.0	2.0	5.0

Table 18. Equipment strength values.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   8725 John J. Kingman Road, Suite 0944
   Fort Belvoir, Virginia 22060-6218

2. Library, Code 52 2
   Naval Postgraduate School
   Monterey, California 93943-5002

3. LTCOL Mark A. Youngren, USA (Code OR/Ym) 15
   Naval Postgraduate School
   Monterey, California 93943-5002

4. Professor Samuel H. Parry (Code OR/Py) 2
   Naval Postgraduate School
   Monterey, California 93943-5002

5. LCDR Jeffrey P. Hamman, USN 1
   1380 Stephens Road
   Virginia Beach, Virginia 23454

6. Deputy Director for Technical Operations, J-8 1
   8000 The Joint Staff
   Washing \ton, DC 20318-8000

7. Chief, Warfighting Analysis Division, J-8 1
   8000 The Joint Staff
   Washington, DC 20318-8000

8. Director, US Army TRADOC Analysis Center 1
   ATTN: ATRC (Mr. Bauman)
   Ft. Leavenworth, Kansas 66027-5200

9. Air Force Institute of Technology 1
   AFIT/ENS
   2950 P Street
   Wright-Patterson, AFB Ohio 45433

10. US Army Concepts Analysis Agency 1
ATTN: Mr. Chandler
8120 Woodmount Avenue
Bethesda, Maryland 20814-2797

11. US Army Concepts Analysis Agency 1
ATTN: Mr. Shepard
8120 Woodmount Avenue
Bethesda, Maryland 20814-2797

12. US Army War College 1
Center for Strategic Leadership
ATTN: COL Wilkes
Bldg. 650
Carlisle, Pennsylvania 17013-5050

13. The Joint Staff - J6AC 1
ATTN: John Garstka
6000 The Joint Staff
Washington, DC 20318-6000